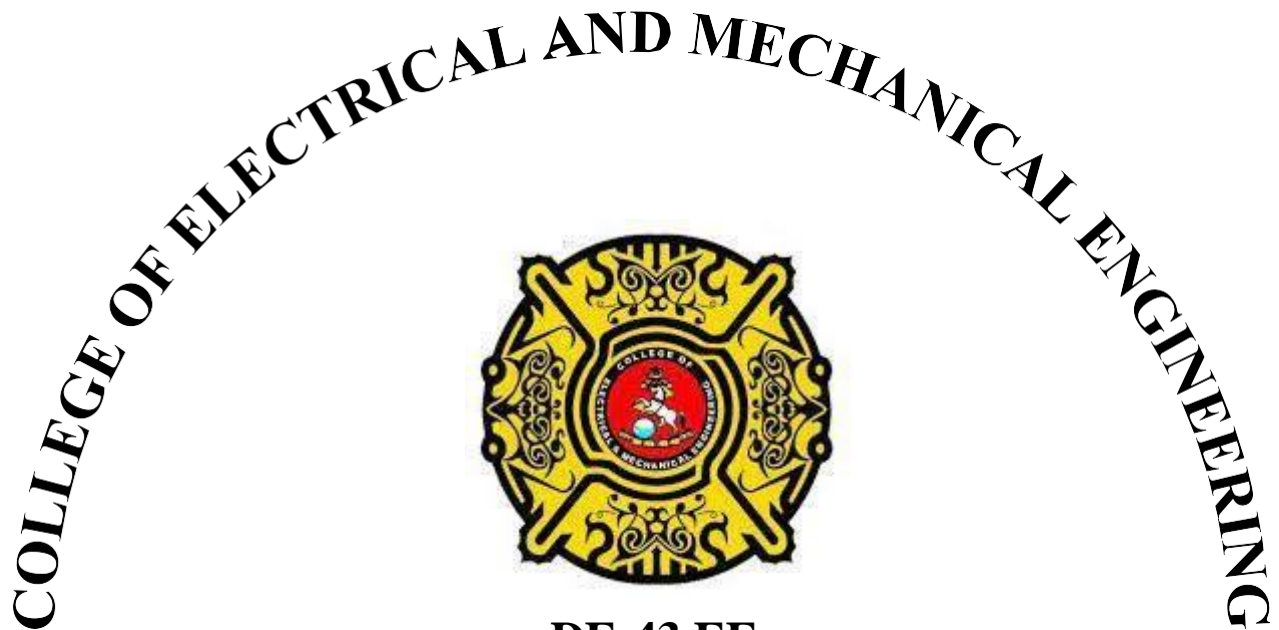


**DE-43 (EE) MUHAMMAD ALI ASIF, MUHAMMAD NABIL AHMAD, SALEEM SHAHZAD**

**REAL-TIME VEHICLE TO EVERYTHING  
COMMUNICATION(V2X) FOR INTELLIGENT  
TRANSPORTATION SYSTEMS**



**COLLEGE OF  
ELECTRICAL AND MECHANICAL ENGINEERING  
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY  
RAWALPINDI  
2025**



**DE-43 EE  
PROJECT REPORT**

**REAL TIME VEHICLE TO EVERYTHING COMMUNICATION(V2X)  
FOR INTELLIGENT TRANSPORT SYSTEM**

Submitted to the Department of Electrical Engineering  
In partial fulfillment of the requirements  
for the degree of  
**Bachelor of Engineering**  
in  
**Electrical**  
**2025**

**Project Supervisor:**

**Lec Furqan Haider**

**Submitted By:**

- 1) Muhammad Ali Asif**
- 2) Muhammad Nabil Ahmad**
- 3) Saleem Shahzad**

## **ACKNOWLEDGMENTS**

With utmost humility, we recognize Allah Almighty's mercy, courage, and intellect by which He blessed us. We have been blessed with such a success within our academic lifetime because of His endless mercies . Alhamdulillah — it was only due to His mercy that we were able to construct the will, endurance, and perception to complete the project.

We would also like to express our sincerest gratitude to our supervisor, the respected Lecturer Furqaan Haider Qureshi, for his support, useful advice, and ever-present encouragement throughout the duration of this project. His guidance and mentorship have played a central role in our work and in overcoming the various obstacles we encountered in the process.

We are very grateful much to the privilege availed by our institution, learning environment, and cooperation of all who assisted us directly or indirectly during this work.

This achievement is not only a testament to our determination and hard work but also to the advice and encouragement given to us by the people around us. May Allah (SWT) grant us more opportunities to learn, grow, and make a positive impact on the world. Ameen.

## **ABSTRACT**

Developed a V2X communication system on LoRa SX1278 modules for real-time data exchange among vehicle nodes, a roadside unit (RSU) proof of concept, and a USRP-based wireless OFDM communication channel for network studies. Established two modes of communications: direct vehicle-to-vehicle (V2V) and Dynamic Source Routing (DSR) for multiple-hop data transfers. In the direct mode, nodes share GPS and acceleration data when in a 3-meter proximity to each other to prevent collisions. Source nodes learn and utilize dynamic routes to forward data to destination nodes. RSSI and SNR values are tracked to measure link quality during the transmission. A Raspberry Pi-driven RSU node executes a YOLO model for traffic congestion detection with video input. The RSU sends congestion information to surrounding vehicles over LoRa for smart driving decisions. A USRP device located at the RSU utilizes another USRP node that acts as the OFDM receiver, demodulating and decrypting the signal to retrieve GPS, acceleration, and traffic information. The system provides an infrastructure for smart, scalable V2X communication, enabling real-time decision-making, improved traffic safety, and seamless integration with future 5G-enabled autonomous transportation networks. OFDM transmitter to broadcast jointly vehicle and RSU data wirelessly over the air.

## SUSTAINABLE DEVELOPMENT GOALS



The Sustainable Development Goals (SDGs) are 17 goals adopted by the United Nations in 2015 as part of the 2030 Agenda for Sustainable Development.

*We aspire to achieve the following SDGs.*



The idea for the project in its core is an inherently innovative solution to an existing problem. The implementation of this idea will lead to a boom in industry as new opportunities will prop up to implement its infrastructure.



Our project plays a pivotal part in establishing smart cities as efficient transportation is an integral part of smart cities. The primary aim of our project is to enhance traffic management, reduce congestion and improve air quality, which are key factors in establishing sustainable cities. It also emphasizes development of smart infrastructure to aid the V2X structure and hence falls in line with this SDG.



Improved facilitated transportation by our efficiency, project, directly contributes to the fulfillment of this SDG. V2X enhances the current transportation structure, causing minimized emissions, and the facilitation of car-sharing development. These advancements promote responsible consumption and production.

# **TABLE OF CONTENTS**

<u>ACKNOWLEDGMENTS</u> .....	i
<u>ABSTRACT</u> .....	ii
<u>SUSTAINABLE DEVELOPMENT GOALS</u> .....	iii
<u>TABLE OF CONTENTS</u> .....	iv
<u>LIST OF FIGURES</u> .....	v
<u>LIST OF TABLES</u> .....	vi
<u>LIST OF SYMBOLS</u> .....	vii
<u>Chapter 1 – INTRODUCTION</u> .....	8
<u>Chapter 2 – BACKGROUND AND LITERATURE REVIEW</u> .....	12
2.1 background .....	14
<u>Chapter 3 – METHODOLOGY</u> .....	17
3.1 hardware components .....	16
3.1.1 Block diagram .....	21
<u>Chapter 4 – RESULTS</u> .....	41
<u>Chapter 5 – CONCLUSIONS AND FUTURE WORK</u> .....	61
<u>REFERENCES</u> .....	62

## **LIST OF FIGURES**

Figure 1 Vehicle Node Architecture.....	9
Figure 2 Esp32.....	15
Figure 3 PCB HARDWARE .....	19
Figure 4 Block Diagram .....	20
Figure 5 Set up for direct Communication.....	20
Figure 6 Receive Packet Function.....	23
Figure 7 Setup for DSR Protocol Implementation.....	28
Figure 8 Setup for OFDM Transmitter Receiver .....	31
Figure 9 OFDM Transmitter .....	32
Figure 10 Stream to Tagged Stream block parameters .....	33
Figure 11 Transmitter block parameters .....	34
Figure 12 OFDM Receiver .....	36
Figure 13 Receiver block parameters .....	36
Figure 14 V2I Block Diagram .....	37
Figure 15 Raspberry pi Hardware .....	39
Figure 16 Channel state information (node 1) .....	42
Figure 17 Channel state information (node 2, node 3, node 4) .....	43
Figure 18 Channel state information( node 2) .....	44
Figure 19 Channel State information (node 3).....	45
Figure 20 Channel State information(node 4).....	45
Figure 21 Node state information (node 1).....	46
Figure 22 Node state information (node 2).....	47
Figure 23 Node state information (node 3).....	48
Figure 24 Node state information (node 4).....	48
Figure 25 Setup for Node 4 with USRP attached.....	49
Figure 26 Data Transferred from Node 1, 2, 3 to Node 4 via Python Script.....	50
Figure 27 Recovered Data stored in a file.....	51
Figure 28 SNR Vs Distance graph .....	52
Figure 29 RSSI Vs Distance graph .....	52
Figure 30 Estimated Distance Vs Actual Distance graph.....	53
Figure 31 OFDM Transmitted Signal.....	54
Figure 32 Tx Spectrum.....	54
Figure 33 Message Signal.....	55
Figure 34 Received OFDM Signal.....	55
Figure 35 Rx Spectrum.....	55
Figure 36 Received Message Signal.....	56
Figure 37 Nodes setup for implementing DSR .....	57
Figure 38 link established between two nodes .....	57
Figure 39 Node 2 matches the destination id.....	58
Figure 40 Route Request is forwarded to Node 3 .....	58
Figure 41 request arrives from multihop at Node 4.....	58
Figure 42 At Node 4, the destination id will be matched.....	59
Figure 43 Node 3 forwarding route reply .....	59
Figure 44 Node 2 forwarding route reply .....	59
Figure 45 Route Reply arrived at Node 1 .....	59
Figure 46 Data packet containing the sensor data .....	60
Figure 47 Dynamic behaviour is verified .....	60
Figure 48 Route Discovered .....	60

## **LIST OF TABLES**

Table 1 lora SX-1278 Parameter .....	13
Table 2 Types of Communication for Sensors .....	16
Table 3 values of parameter .....	21
Table 4 loRa Funtions .....	22
Table 5 LoRa Main Funtions .....	24
Table 6 USRP RF settings .....	35



## **LIST OF SYMBOLS**

### **Acronyms**

V2X	Vehicle to Everything
V2P	Vehicle to Pedestrian
V2N	Vehicle to Network
V2I	Vehicle to Pedestrian
USRP	Universal Software Radio Peripheral
RSU	Road Side Unit
RSSI	Receives Signal Strength Index
SNR	Signal to Noise Ratio

# **Chapter 1 - INTRODUCTION**

## **1.1. Background and Motivation**

The V2X (Vehicle-to-Everything) project is motivated by the desire to revolutionize the face of transportation. Fundamentally, the project aims to revolutionize road safety through the application of sophisticated communication technologies such as cellular networks and an extensive understanding of communication protocols. Additionally, V2X promises to improve traffic efficiency and optimization through smoother traffic flow, congestion reduction, and route optimization through cooperative driving functions such as platooning and adaptive cruise control. In addition to safety and efficiency, the project also aims to provide environmental benefits by reducing fuel consumption, greenhouse gas emissions and air pollution by rendering transport across roads more energy-efficient.

Also, the V2X project is motivated by the vision of providing users with greater mobility, accessibility, and diversity. Through the provision of real-time information and warning systems, V2X technology allows pedestrians, cyclists, and mass transit vehicles to safely and efficiently move along roads. Inclusive functionalities like audible warnings and tactile signals also enhance inclusivity since people with disabilities can independently use transport facilities. V2X also provides a foundation for the creation of autonomous driving technology, which allows vehicles to sense and react to their environment without human control. With its adoption of technological innovation and cooperation, the V2X the project seeks to develop economic benefits through the elimination of the economic cost linked with traffic accidents, costs due to congestion, and wasteful transport systems, thus contributing to economic development, employment opportunities, and greater transport competitiveness.

## **1.2. Problem Statement**

Safety on roads is an issue in modern transport networks, with crashes and accidents perpetual risk to drivers, foot passengers, and assets surrounding them. Also, inefficiency in communication between infrastructure and vehicles makes it harder to deal with these challenges, hindering speedy detection and response to risks. There is a pressing need for the deployment of vehicular communication systems that can enhance road safety by providing a remedy to the issues mentioned above.

## **1.3 Project Objectives**

1. Create a low-cost, hardware-based vehicle-to-vehicle communication proof-of-concept using ESP32 and LoRa to facilitate real-time data sharing between on-the-move vehicles without the need for external infrastructure.
2. Deploy LoRa modules to enable energy-efficient, long-range communication between vehicles, such that the system can be implemented in different road conditions, like highways and urban setups
3. Use MPU6050 (accelerometer and gyroscope) and GPS sensors to accurately detect vehicle movement, positioning, and orientation to support context-aware communication and enhance situational awareness.
4. Enable vehicles to communicate directly with each other using custom-designed protocols inspired by ad hoc routing (e.g., DSR), promoting a fully decentralized system that operates independently of the internet or cellular networks.

### 1.3 Design Workflow

Our project design process was specifically planned to develop a systematic and effective development process for a secure V2X communication system. We started with the definition of the project scope, the critical communication scenarios like Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I), and wireless protocols like DSRC and LoRa. We then designed the hardware and software architectures concurrently, after requirements analysis. On the hardware platform, we brought together modules including the ESP32 microcontroller, LoRa transceivers, GPS (NEO-6M), and the MPU-6050 sensor to create the heart of our embedded system. At the same time, the software team prepared the firmware that would enable the acquisition of sensor data, distance estimation based on path loss, and real-time LoRa communication. Simulations were done utilizing MATLAB and GNU Radio with USRP to emulate the communication scenario and test performance under different circumstances. After testing individual modules, we progressed towards system integration, and subsequent field testing for measuring the robustness of the system in dynamic driving conditions. The process was iterative, enabling frequent feedback and adjustments based on real-world performance.

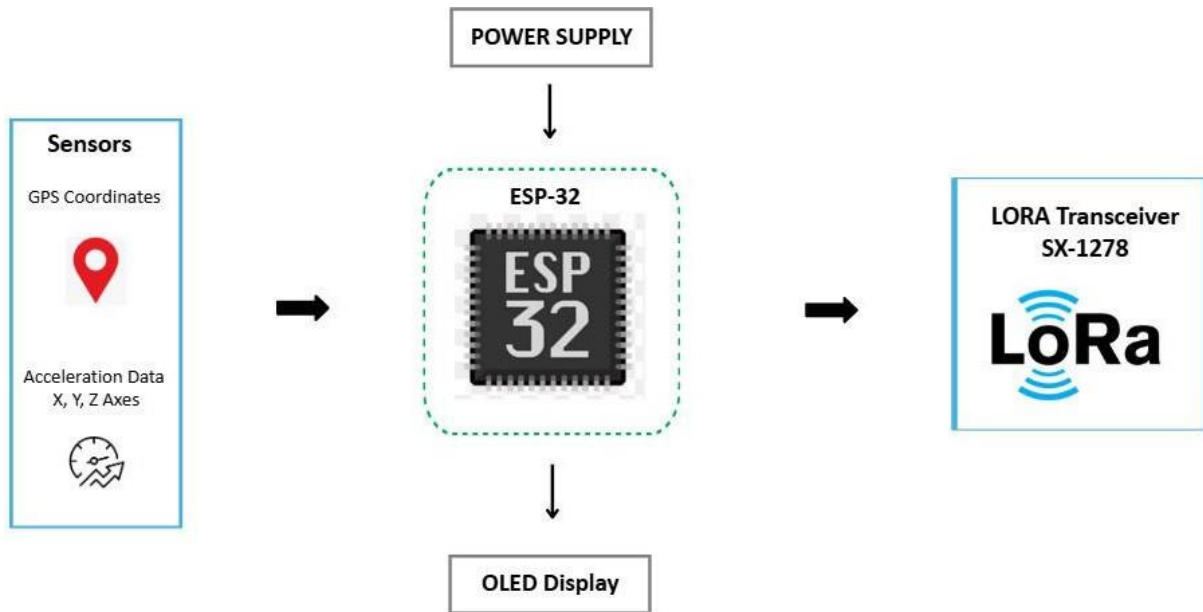


Figure 1 Vehicle Node Architecture

#### 1.3.1 Vehicle-to-Infrastructure (V2I)

**Vehicle-to-Infrastructure (V2I)** permits communication between vehicles and roadside infrastructure with the goal of enhancing traffic flow and safety.

- **Infrastructure Side**

Vehicle detection and counting is performed by a Raspberry Pi using YOLOv8. Congestion levels (Low / Medium / High) are estimated based on the density of vehicles in a specific Region of Interest (ROI).

- **Communication**

Congestion data is sent to an ESP32 microcontroller through serial communication. Using LoRa, a low-power long-range wireless protocol, the ESP32 transmits this data to vehicles within range.

- **Vehicle Side**

Vehicles receive congestion information which can be used to change the intended route or driving behavior accordingly.

## **1.3.2 Introduction to YOLO and YOLOv8**

### **1.3.2.1 What is YOLO?**

**You Only Look Once (YOLO)** is an exceptional-based algorithm for object detection that recognizes objects in pictures through real-time detection in a single-step neural network process. Unlike prior models like R-CNN that depend on region proposals, YOLO takes a more holistic view. Firstly, it splits the input image into grids and aggregates the following predictions:

- Confidence scores
- Class probabilities
- Bounding boxes

Due to the effectiveness of YOLO and its low latency, it's popularly used in autonomous systems, surveillance, and traffic control.

### 1.3.2.2 Working of YOLOv8

From **Ultralytics**, YOLOv8 is the most recent version with more improvements to the architecture aimed at boosting speed, precision, accuracy, and scalability. Important highlights are:

#### 1) Anchor Box-Free Detection

- No longer requires set anchor boxes
- Model significantly streamlined
- The object's centroid and dimensions are predicted directly

#### 2) CSPDarknet Backbone

- Extracting of features is done better using cross-stage partial connection of CSP with SiLU (Swish-like) activation functions

#### 3) Multi-Scale Feature Fusion

- Detection performance of smaller and bigger objects enhanced by the incorporation of a PANet-like neck for fusion of features from varied levels

#### 4) Task-Aligned Assigner

- By improving the alignment of classification and localization tasks for tuning, training is better aligned with the defined objectives

In this project, YOLOv8 processes video frames to detect vehicles (e.g., cars, trucks, motorcycles) and counts them to estimate traffic congestion. Its lightweight architecture allows deployment on resource-constrained devices like the **Raspberry Pi**, ensuring real-time inference without compromising accuracy

## 1.4 Project Benefits

### 1. Infrastructure-Free

The platform is fully functioning without the use of roadside units, cellular towers, or cloud servers, a fact that enhances its suitability to remote and undeveloped areas lacking communication infrastructure or where no communication infrastructure can be found.

### 2. Affordable and Scalable

With low-cost components such as ESP32 and LoRa used, the solution keeps production costs low with simple scalability in use over great numbers of motor vehicle fleets or public transport schemes.

### 3. Road Safety Improvement

Exchange of motion and location data between vehicles in real-time can help provide early warning of impending collisions, hard braking, or hazardous road conditions, eventually resulting in safer driving conditions.

### 4. Energy-Efficient Design

Low energy consumption by the system due to the use of LoRa and low-power microcontrollers makes it ideal for constant use in vehicles without overloading their power systems

## **Chapter 2 - LITERATURE REVIEW**

### **2.1. Introduction to V2X Communication**

Vehicle-to-Everything (V2X) communication represents the future of road transport, aiming to enhance security with a reduction in accidents and improvement in traffic flow. It will also enable real-time sharing of data between automobiles and other elements in their environment. It will include other vehicles (V2V), infrastructure (V2I), or even broader networks. With advancements now cities are becoming smarter, and communication systems have played a significant role in traffic automation and accident prevention.

Due to many modifications in V2X, instead of relying on heavy infrastructure or centralized systems, the modern approach is shifting towards lightweight and embedded solutions. It will allow vehicles to communicate autonomously, even in remote areas. Our project represents this idea so by implementing this, we can approach a lightweight and real-time V2X communication model. It is beneficial due to its cost-effective hardware and efficient communication protocols.

### **2.2 Direct Communication in V2V Systems**

Direct communication allows vehicles to talk to each other without relying on intermediate infrastructure. It will be especially useful in areas with weak cellular coverage or no fixed roadside units. In critical scenarios like emergency braking or obstacle warnings, direct communication reduces latency, which will make it a preferred method for real-time vehicular applications.

In contrast to centrally based systems that potentially cause delays because of routing and processing overhead, direct peer-to-peer communication will provide more rapid responses. This will be imperative for safety applications where even a few tenths of a second could be the difference between a near miss and a collision

### **2.3 Dynamic Source Routing (DSR) in Vehicular Networks**

Dynamic Source Routing (DSR) has become one of the most foundational protocols used in mobile ad hoc networks (MANETs), and its adaptability has made it a strong candidate for vehicular network applications. In the context of vehicle-to-vehicle (V2V) communication, DSR has allowed each vehicle to determine and maintain the route it uses to send messages by embedding the entire path in the packet header. This approach has offered flexibility, especially in dynamic environments where network topology changes rapidly—something that's very common on roads.

One of the key strengths is its topology which includes on-demand route discovery processes.

Vehicles have not need to maintain constant route tables, which reduces memory and processing overhead. When a message needs to be sent, the vehicle initiates a route request, which then travelled through the network until it finds the destination. Once the destination had been reached, a reply could sent back using the discovered path. This method has ensured efficient communication in environments where vehicle positions are constantly changing.

DSR also has its limitations. In high-mobility scenarios like highways or urban traffic, routes could become outdated could despite lead to increased overhead due to frequent rediscovery attempts.

Despite this, DSR has remained a widely studied protocol due to its simplicity, ease of implementation, and suitability for scenarios where network infrastructure has absent or unreliable. In our project, DSR-inspired logic supports the communication structure, which has allowed each vehicle to function independently without needing central control. It has further strengthened the

decentralized and scalable nature of the proposed system.

## 2.4 Orthogonal Frequency Division Multiplexing (OFDM)

OFDM is a digital modulation technique used in numerous wireless network architectures, is used to transmit data at a very high rate while utilizing the bandwidth efficiently. In this technique, bandwidth is divided into different subcarriers, which are transmitted simultaneously. This is the key technology behind the WIFI, 4G, and 5G systems.

### 2.4.1 Working Principle

Inverse Fast Fourier Transform is used to convert the frequency subcarriers into a time domain signal carrying the modulated data. The time domain signal is then transmitted by the radio hardware and can be demodulated at the receiver end. At the receiver, the Fast Fourier Transform is applied in order to extract the original subcarriers back which contain the modulated data. Different modulation schemes, including QPSK, BPSK, etc. are used to modulate the data and recover it at the receiver.

### 2.4.2 OFDM in V2X Communication Systems

Real-time data exchange for use in applications such as autonomous driving, traffic management, and safety features is required in V2X communication. It can enable efficient and reliable communication between vehicles, infrastructure, and roadside units.

## 2.5 LoRa

LoRa (Long Range) has a wireless modulation technique derived from Chirp Spread Spectrum (CSS) technology. It is widely recognized for enabling long-range, low-power communication, making it a wonderful choice for V2X applications in environments where traditional DSRC or LTE-V2X may be limited due to cost or infrastructure constraints. In a typical V2V communication setup, vehicles will need to maintain connections over several hundred meters or even kilometers. LoRa has operated in unlicensed spectrum bands like 433 MHz or 868 MHz, enabling data transmission across long distances with relatively little interference. Moreover, its low energy footprint made it suitable for battery-powered embedded systems used in vehicles.

Specification	Details
Operating Frequency	433 MHz
Bandwidth	125 kHz
Range (LOS)	15 km in rural areas 2–5 km in urban areas
Range (NLOS)	2–5 km in rural settings 1–3 km in urban regions
Sensitivity	Up to -148 dBm
Operating Voltage	1.8V – 3.7V

*Table 1 lora SX-1278 Parameter*

Compared to more complex wireless standards, LoRa has offered a simpler and cheaper alternative for the transition of location and security-related information between vehicles.

## **2.6 Spread Spectrum Technique in LoRa Communication**

### **2.6.1 Introduction to Spread Spectrum**

Spread Spectrum is a modulation technique which spreads the signal over a wide frequency spectrum, and which is less susceptible to interference, noise, and signal jamming. This technique enables low power long range communication, within minimum end to end delay, which is required V2X communications.

LoRa SX-1278, used in our project uses an implementation of Chirp Spread Spectrum (CSS) to accomplish low-power and long-distance communication.

Following are the key configuration parameters of Spread Spectrum technique for communication:

### **2.6.2 Spreading Factor (SF)**

A chirp signal used in LoRa modulation to encode data, is a signal in which frequency increases or decreases linearly over time. This change in frequency makes the signal more resilient to noise and interference, over long ranges. Number of Chirps used per symbol are controlled by spreading factor. A greater value enables long range communication while compromising the data rate. A moderate value of this factor is 7, commonly used in many applications.

### **2.6.3 Coding Rate (CR)**

This factor enables to send data correction bits in addition to the original data bits in order to correct any errors in the transmission. High coding rate enhances the error correction but reduces the data rate. A factor suitable for the network is set at which the receiver and transmitter are synchronized at required data rate.

### **2.6.4 Sync Word**

Explanation: Sync word helps the receiver to identify the start of the packet. Its like a header used to synchronize the receiver and transmitter accurately. It avoids the interference from other devices and prevents the receiver to catch data packets from other networks working in the same frequency band.

### **2.6.5 Cyclic Redundancy Check**

CRC is a method which detects the errors by checking the checksum added by the transmitter at the receiver end. The received data packet is considered valid only if the checksum is matched otherwise an error is detected. Reliable communication especially in case of noisy environments is ensured by adding this factor in the data.

## **2.7 ESP32**

At the middle of our communication nodes has the ESP32 microcontroller. It has a dual-core, low-power chip with integrated Wi-Fi and Bluetooth. As Wi-Fi and Bluetooth are not the main communication tools in the project, the ESP32's ability is to handle multiple processes and deal



with peripherals that made it a perfect choice.

It has acted as the system's brain, managing sensor data, processing location and movement information, and broadcasting it via the LoRa module. One of ESP32's biggest advantages has its compatibility with open-source development environments, made it rapid prototype and deployment much easier.



*Figure 2 Esp32*

## **2.8 GPS and Motion Tracking for Real-Time Alerts**

In a V2X system, it has not enough to just know that another vehicle is nearby—you need to know where it is, how fast it's going, and whether it's about to stop or turn. That is place where GPS and motion sensors like the MPU6050 come in.

The GPS module have provided accurate real-time location data. It has allowed vehicles to determine their own position and receive positions of nearby vehicles that will enable decision-making based on spatial awareness. The MPU6050 sensor has combined a 3-axis accelerometer and a 3-axis gyroscope, which detect vehicle movement. It could be used to recognize sudden braking, sharp turns, or possible collisions.

This data, when shared with other vehicles using LoRa, will form the basis for intelligent warnings. For instance, if one vehicle brakes suddenly, nearby vehicles receive an immediate alert and could take action accordingly.

*Table 2 Types of Communication for Sensors*

Sensor / Module	Function	Communication Type
<b>MPU6050</b>	Accelerometer + Gyroscope (Motion Detection)	I2C
<b>NEO-6M GPS Module</b>	Global Positioning System (Location Data)	UART
<b>LoRa SX1278 Module</b>	Long-Range Wireless Communication	SPI (typically)
<b>ESP32</b>	Microcontroller + Wi-Fi/Bluetooth	Multiple: UART, I2C, SPI (depending on connections)
<b>USRP (NI/Ettus)</b>	Software Defined Radio (for analysis)	USB / Ethernet (depends on model)

## 2.9 Related Work and Gaps in Literature

Several researchers had explored vehicular communication using expensive solutions like LTE or 5G, which require mobile networks and carrier services. While others will rely on complex routing protocols or infrastructure that may not be available in rural or developing regions.

Very few studies are focusing on the implementation of V2X communication using LoRa, despite its advantages in range and simplicity. Additionally, many existing projects did not consider vehicle motion data in real time, focusing only on static location or speed. Our system is closing this gap by combining both GPS and motion tracking for a more comprehensive awareness solution.

## 2.10 Novelty and Impact of the Proposed Work

Our approach has been differentiated because it is simple, cost-effective, and applicable to real-life situations. It eliminated the need for external infrastructure and can work independently in rural settings. Direct communication via LoRa has ensured data exchange in real-time, with ESP32, GPS, and MPU6050 providing real-time processing, location, and motion analysis.

Our system can be especially useful in third-world countries where roads are often unpatrolled and accidents are common due to ignorance. By enabling cars to "speak" to each other, our model enhanced safety, reduced collisions, and established the potential for intelligent, networked transportation.

## **Chapter 3 – METHODOLOGY**

### **3.1 Hardware Components**

#### **3.1.1 Microcontroller Unit ESP32**

ESP32 microcontroller is the main processing device of each node in our communication system. In our project, it is being used for communicating with the sensors, LoRa module, and OLED display via communication protocols.

#### **3.1.2 The Role Played by the LoRa Module**

##### **A. Introduction to LoRa Communication**

LoRa (Long Range), a low-power long-range wireless communication Chirp Spread Spectrum (CSS) modulation technology.

##### **B. Setting Up the Antenna and Hardware Interface**

The LoRa SX1278 module used in this project features an IPEX antenna, a tiny radio frequency link that strengthens signals and ensures reliable wireless communication.

- The IPEX connector routes the RF signal from the transceiver to the external antenna.
- This setup boosts the communication range and signal clarity, especially in mobile environments.

##### **C. Transmission Process**

Data Preparation on ESP32: Real-time acceleration and position data, is collected by ESP-32 from the sensors which are MPU6050 (for motion and acceleration data) and GPS (for coordinates).

**Protocols for communication with ESP-32 :** I2C(Inter Integrated Circuit) protocol is used for the MPU6050 and OLED, UART(Universal Asynchronous Receiver/Transmitter) protocol for the GPS sensor. These are standard communication protocols used for sensors integration with microcontroller.

To guarantee dependable transmission over the wireless channel, converted into a structured string or byte stream, and encoded. This encoded data is then forwarded to LoRa module.

**SPI Communication with SX1278:** For interfacing LoRa with ESP-32, SPI(standard peripheral interface) communication protocol is used.

The frequency , bandwidth , transmission power, and spreading factor are initialized to the internal registers of the LoRa module by ESP32. After these initial parameters are set, structured and encoded data is transmitted from the ESP32 to the SX1278 LoRa transceiver using the SPI interface pins allocated in the code.

CSS modulation is applied on this incoming data which enables long range communication. The modulated signal is then amplified and sent through the IPEX antenna attached to the module. The radio frequency (RF) signal is broadcasted wirelessly, reaching nearby vehicle nodes (OBUs) or the

Road Side Unit (RSU) node, depending on the network configuration.

## **D. Reception Process**

The incoming RF signal transmitted is captured by IPEX antenna over the air. This received signal is then forwarded to the SX1278 transceiver for further processing. Using Chirp Spread Spectrum (CSS) technique, the received signal is demodulated by the SX1278 module, which helps in maintaining low latency and resistance to interference.

Data Transfer to ESP32:

After successful demodulation, the payload is sent to the ESP32 via the SPI (Serial Peripheral Interface) pins defined in the code. The ESP32 parses the incoming data and separates relevant fields (e.g., GPS location, acceleration, or sender ID). Based on the data received, the ESP32 can trigger alerts, updates OLED displays, or stores the information for further use in real-time decision-making (e.g., congestion detection, obstacle alerts).

### **3.1.2 Roadside Unit (RSU) Architecture**

#### **3.1.2.1 Hardware Components**

- Raspberry Pi 3
- ESP32
- LoRa SX1278 Transceiver (as previously described in Vehicle Nodes)
- USRP B200
- VERT 900 Antenna

#### **• 3.1.2.2 Traffic Congestion Detection**

- Video Input → Raspberry Pi → Vehicle Count → Congestion Level
- Congestion Level Transmission via LoRa Network to Other Vehicles

#### **• 3.1.2.3 OFDM Transmission and Reception**

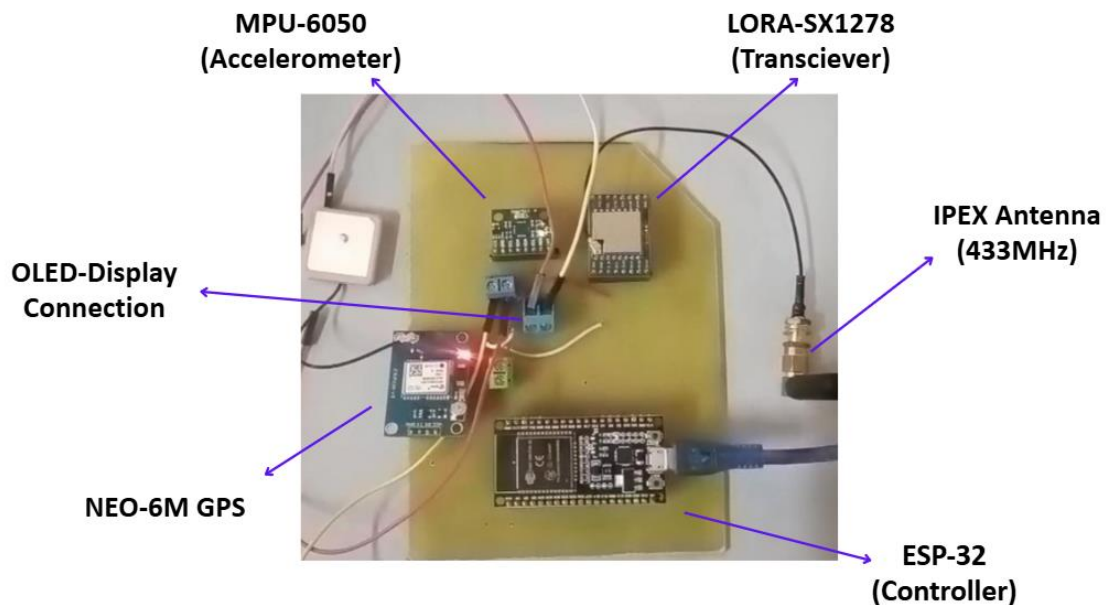
At the RSU Node, the data is received from the vehicle nodes (nodes 1, 2 and 3) using the LoRa network. The received data is displayed in the Arduino IDE software. This data is using the OFDM transmitter block in GNU Radio, modulated, and transmitted over the air from USRP. At the receiver side, another USRP connected with a laptop receives this OFDM signal, demodulates it, and recovers the original payload data.

The objective of this setup on USRPs is to perform OFDM signal analysis, demonstrating the efficient recovery of the data file and analyzing the spectrums of transmitter and receiver.

- OFDM Transmitter using GNU Radio Companion and USRP B200
- Data Input: Connection bw Arduino IDE and GNU Radio
- OFDM Transmission Over the Air
- OFDM Receiver on Second USRP and GNU Radio for Demodulation
- Data Recovery and Display on Connected Laptop

## PCB Implementation

We have implemented this circuit on a PCB FR-4. In total, we have fabricated three PCB units, one for each vehicle node.



*Figure 3 PCB HARDWARE*

### 3.2 BLOCK DIAGRAM

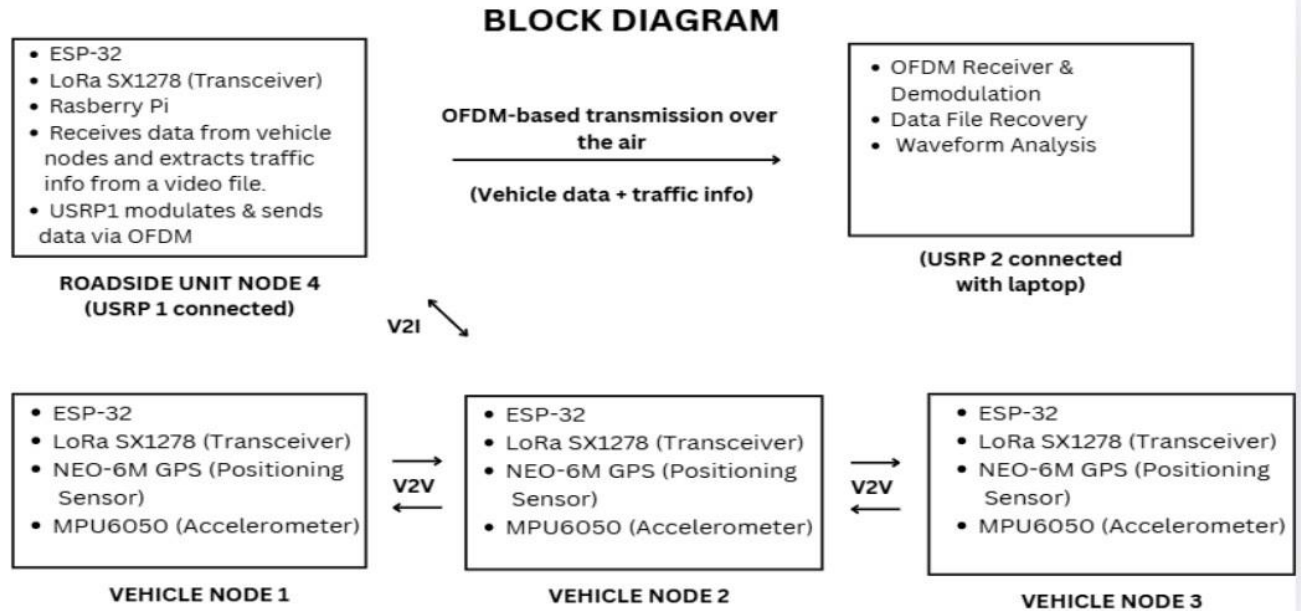


Figure 4 Block Diagram

### 3.3 Communication Modes

#### 3.3.1 Direct Communication Mode

All the four nodes are placed at a distance apart, each connected to a laptop with each node displaying its status and channel information on the Arduino IDE software.



Figure 5 Set up for direct Communication

## i. LoRa Communication Parameters Configuration

In the start of code for each node, we have set up the parameters as of loRa network. Below is the code for parameters setting:-

```
// Set the LoRa parameters
LoRa.setSpreadingFactor(7);
LoRa.setSignalBandwidth(125E3);
LoRa.setCodingRate4(5);
LoRa.setSyncWord(0x12);
LoRa.enableCrc();
```

Following is the table explaining the values set of each parameter:

Parameter	Setting	Description
<b>Spreading Factor</b>	7	Provided a suitable tradeoff between data rate and range.
<b>Signal Bandwidth</b>	125 kHz	Reasonable required range was acquired by setting this bandwidth.
<b>Coding Rate</b>	4/5	Error correction level is set, to improve the reliability of data transmission even in noisy environments
<b>Sync Word</b>	0x12	Unique identifier for the LoRa network in order to prevent interference from other networks. Packets only with same sync word are received.
<b>CRC (Cyclic Redundancy Check)</b>	Enabled	Enabled to detect errors in transmission.

*Table 3 values of parameter*

## ii. Beacon Message Broadcasting

Each vehicle node will broadcasts a beacon message, which will be received by all nearby nodes including vehicle and roadside unit. This initial message will establish a link between all nodes, which will further be used for examining the distance and other parameters such as SNR, and packet size. Below is the code for message broadcasting:-

In the code, periodically after 10 seconds, a function named sendPacket() is being called. In this function, a Lora packet is broadcasted as follows:

```
void sendPacket() {
  // Send a packet with the current node's name
  LoRa.beginPacket();
```

```
String packetText = "Beacon Message from Vehicle NODE 1";
LoRa.print(packetText);
LoRa.endPacket();

Serial.println("\nSent packet: " + packetText);
}
```

Function	Description
<b>LoRa.beginPacket()</b>	Initializes the internal buffers of the LoRa module to store data to be sent, sets up the appropriate communication settings defined in code (frequency, spreading factor, bandwidth, etc.), to send the data.
<b>LoRa.print(packetText)</b>	Data is sent into the transmission buffer by this line.
<b>LoRa.endPacket()</b>	Buffered data is sent over the LoRa network.

*Table 4 loRa Funtions*

From this received beacon message packet, signal strength, signal to noise ratio, and size have been measured and displayed for real-time network assessment.

### iii. Processing Received Data

The module remains in receive mode for a predefined duration specified by the constant RECEIVE\_DURATION in the code, which is set to **10 seconds**. During this time, the module listens for incoming packets from other nodes.

The module remains in receive mode for 10 seconds defined in the code by a constant named RECEIVE\_DURATION. Periodically in the loop() function of code, a function named receivePacket(RECEIVE\_DURATION); is called, with RECEIVE\_DURATION named passed as an argument. LoRa.parsePacket() function is used to check if any packet is received. If no valid packet is found, it returns zero, otherwise, it reads the data by the following code:

```
if (packetSize) {
  String receivedText = "";
  while (LoRa.available()) {
    receivedText += (char)LoRa.read();
  }
}
```

The RSSI and SNR are extracted from the received packet as follows:

```
int rssi = LoRa.packetRssi();
float snr = LoRa.packetSnr();
```



After decoding, the data is processed and relevant information such as **node name**, **payload**, **RSSI**, and **SNR** are extracted.

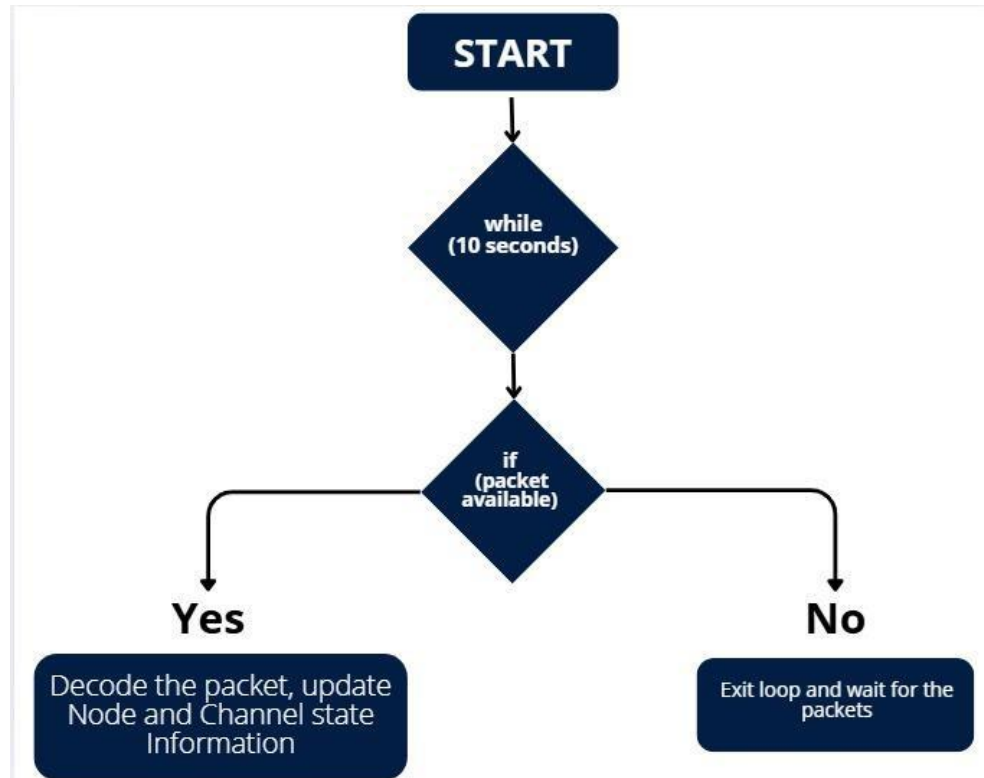


Figure 6 Receive Packet Function

#### iv. Channel State Information

At each node, a beacon message is received, and channel state information is estimated from this message. The signal-to-noise ratio and received signal strength are measured from pre-defined libraries in Arduino IDE software. `LoRa.SNR()` and `LoRa.RSSI()` are the functions that return the respective values of SNR and RSSI(received signal strength indicator).

Function	Description
<b>LoRa.SNR ()</b>	Initializes the internal buffers of the LoRa module to store data to be sent, sets up the appropriate communication settings defined in code (frequency, spreading factor, bandwidth, etc.) to send the data.

<b>LoRa.RSSI ()</b>	Data is sent into the transmission buffer by this line.
<b>LoRa.parsePacket ()</b>	The received packet size is stored in an internal buffer. LoRa module reads this size from the buffer internally at the backend and returns the size (including payload and header).

*Table 5 LoRa Main Functions*

#### **LoRa.RSSI () function:**

The received **analog RF signal** is first demodulated to extract the power level, using an internal power detector. This detector measures the voltage fluctuations of the signal which are directly related to power. The RSSI value is then calculated by this power level, by comparing it to 1mW(0 dBm) power by following the formula:-

$$\text{RSSI (dBm)} = 10 \times \log_{10} \left( \frac{\text{Received Power (in mW)}}{1 \text{ mW}} \right)$$

#### **LoRa.SNR () function:**

After the Radio Frequency signal is received, Chirp Spread Spectrum (CSS) demodulation is applied, detecting the original signal's chirp pattern, while rejecting the noise. Formula for calculating SNR is:

$$\text{SNR (dB)} = 10 \times \log_{10} \left( \frac{\text{Signal Power}}{\text{Noise Power}} \right)$$

### **v. Node State Information**

In the Node State Information, real-time status updates for each node in the network, are displayed such as the data received, the signal strength and Signal to Noise Ratio of received signal, and the number of packets received from a particular node.

## 1. Node Data Reception

Inside the receivePacket() function, the received data, signal strength and signal to noise ratio are calculated and displayed . This information is then updated for that specific node by the function named updateNodeInfo().

## 2. Node Status Updates

Inside the receivePacket() function, a packet is successfully detected, the node updates and displays the following information:

- **Signal Strength:** RSSI of the received signal is evaluated for further analysis.
- **Distance Estimate:** The approximate distance of this node is displayed.
- **Packet count:** This shows the number of packets received from this particular node.

## 1. Distance Estimation

The **distance** between nodes is estimated based on the **RSSI (Received Signal Strength Indicator)** value of the received packet using the **path loss model formula**:

$$Distance = \frac{10^{(A-RSSI)}}{10 * n}$$

where,

n = path loss constant = 2 (for Line of Sight)

A = RSSI value at 1 meter distance = -40dBm in our case

RSSI = strength of received packet

**Nearest Node Determination:**

Nearest node is determined from the function `determineNearestNode()` while displaying the node state information data.

It iterates over all the nodes, checks if data is received from that node, compares all available distances and finalizes the nearest node.

```
if (nodes[i].dataReceived && nodes[i].distance > 0 && nodes[i].distance < minDistance) {  
    minDistance = nodes[i].distance; // Update the nearest node  
    nearestIndex = i;  
}
```

## 2. Data Exchange Logic

Inside `determineNearestNode()` function, after determination of minimum distance and nearest node, there is an if condition which checks if the distance of a node is less than **2 meter** range, it captures Sensor Data(GPS Coordinates, and Acceleration) and unicasts a datapacket for that node.

```
// If a node is found within 2 meters, capture and send sensor data  
if (nearestIndex != -1 && minDistance < 2.0) {  
    captureSensorData(nodes[nearestIndex].name);  
}
```

This distance based sharing of sensor data reduces network congestion and shares the critical data only when a vehicle is nearby. Node's power is conserved and reliable communication is ensured by limiting data transmission over short distances only.

## 3. Data Acquisition from NEO-6M GPS and MPU6050 Sensors

In the `captureSensorData()` function, serial communication techniques, UART (Universal Asynchronous Receiver/Transmitter) for the GPS and I2C (Inter-Integrated Circuit) for the MPU6050 sensor are used to communicate the data to ESP32.

### NEO-6M GPS Communication:

The GPS data is communicated using UART, where the GPS module sends serial data to the GPIO 16 pin of ESP32 used as receive pin for UART communication.

GPS data is decoded by the TinyGPS++ library as follows:

```
while (gpsSerial.available() > 0) {  
  
    char gpsData = gpsSerial.read();  
  
    GPS.encode(gpsData); // Feed the GPS data to TinyGPS++  
  
}
```

Byte-by-byte GPS data is read by `gpsSerial.read()` function. This binary format data is processed by the instance of TinyGPS++ defined in the code, such as:

- `gps.location.lat()`: returns Latitude value
- `gps.location.lng()`: returns Longitude value

### **MPU6050 Communication:**

The MPU6050 sensor communicates with the ESP32 over the I2C protocol.

After initialization, the data is fetched by the `mpu.getEvent()` function from the MPU6050 sensor. This function reads accelerometer values from the sensor as follows:

```
sensors_event_t accel, gyro, temp;
```

```
mpu.getEvent(&accel, &gyro, &temp);
```

where,

- `sensors_event_t` is a struct defined in the Adafruit MPU6050 library to store sensor data.
- `mpu` is the instance of Adafruit library defined in code
- `accel, gyro, temp` are objects made to store the data

From these acceleration values are obtained:

`accel.acceleration.x` gives X-axis acceleration

`accel.acceleration.y` gives Y-axis acceleration

`accel.acceleration.z` gives Z-axis acceleration

This data represents the movement or tilt of the sensor (placed on the vehicle node's PCB) in three-dimensional space.

### **Packet Formatting:**

The data is then formatted into a packet string that contains information about the source node, destination node, and payload (which is the combined GPS and accelerometer data).

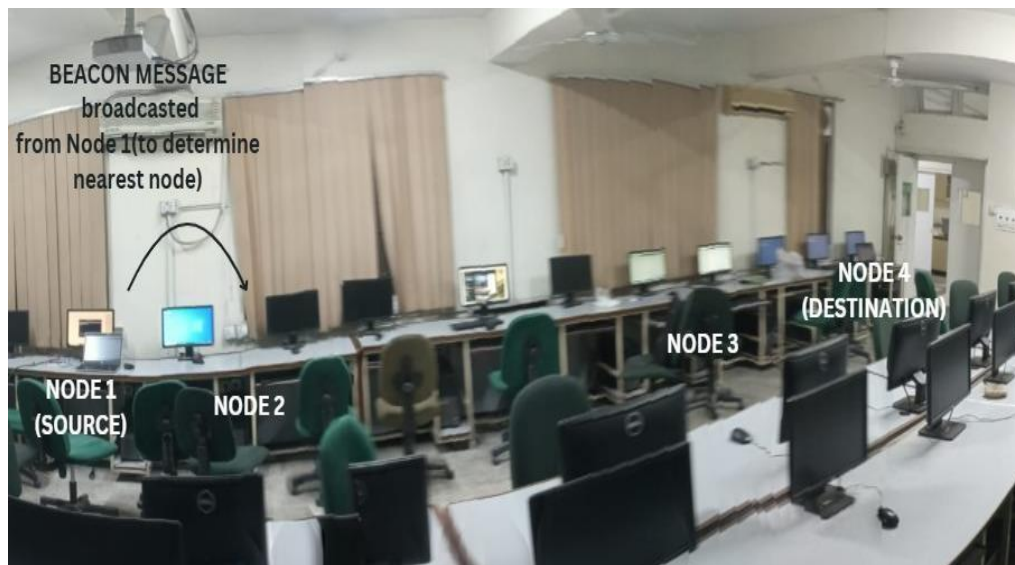
The data is then formatted into a packet and unicast to the destination node (node within a 2-meter vicinity). This packet contains header and payload data as follows:

**DATA: SOURCE: NODE 1: DESTINATION:<Nearest Node>: PAYLOAD:<GPS and Accelerometer Data>**

On the receiver side, the receiver node matches the destination with its own name id, if matched then receives and displays the data otherwise discards it.

### **3.3.2 Dynamic Source Routing (DSR) Protocol**

The objective of this protocol is to determine a route from the Source node to the Destination Node. Node 1 is designated as the Source node, **Nodes 2 and 3** are **intermediate nodes**, and Node 4 is the destination node. Once the route is determined, the captured sensor data from the source node is transmitted to the destination node. Packets are relayed from hop to hop, processed, and forwarded by each node is displayed. Dynamic behavior is ensured by changing the location of nodes and verifying whether or not the change in route is detected.



*Figure 7 Setup for DSR Protocol Implementation*

## 1. Route Discovery

Node 1 broadcasts a beacon message to its nearby nodes periodically until a beacon reply is received. Its nearest node, i.e. Node 2, catches this signal and generates a beacon reply.

```
// sending beacon messages
if (!nearestNodeDetermined) {
    sendBeaconMessage();
    delay(4000);
}

void sendBeaconMessage(){

    LoRa.beginPacket();
    LoRa.print("BEACON:" + currentNodeName);
    LoRa.endPacket();
    Serial.println("BEACON msg sent successfully " ); // above loop executed
}
```

The packet format for the beacon message is:

**BEACON: Node1**

### Beacon Message Reception:

The nearby node of Node 1, which in the above figure is Node 2, will receive the beacon message. Upon receiving it, it processes it and generates a beacon reply.

The packet format for beacon reply is:

**BREPLY:PacketDest:sendernode:currentNodeName**

Node 1 which has initiated the beacon message will extract the node name from this beacon reply, and measure distance based on its received signal strength RSSI. In the code, there is some delay added after receiving the beacon reply so that if in the meantime another node replies to the beacon message its distance is also calculated. Based on the beacon reply and distance first node in the route discovery is determined.

### Route Request

After the nearest node is determined by Source node 1, a Route Request packet is initiated by it. This

packet is unicasted only to the nearest node, with its name after the packet destination. The nearest node which is Node 2 will receive this packet and send back an acknowledgment signal to confirm that the request signal has been received.

Since Node 2 is not the destination, it will append its name in the route and forward this request to its next nearby node. The process of finding the nearest node at Node 2 will begin after it has received the route request. It will again in a similar way broadcast a beacon message, receive the beacon message replies, and based on distance determine its nearest node. After the nearest node determination by Node 2, it will append its name in the Route of Request packet and forward the request to its nearest node as follows:

**RREQ: PacketDest: <nearestNode>: <source>: <destination>: <Node1 Node2>**

- <nearestNode> is the nearest node determined.
- <source> is the source node which initiates this request.
- <destination> is the final destination node in the route.
- <route> is the actual route to find the destination node

In the same way, Node 3 receives the request packet, finds its next nearby node, appends its name in the route, and forwards this request to the next node. The next node after Node 3 is the fourth node which is the destination. Finally, the route request reaches destination Node 4.

**Figure** Block diagram showing Hop to Hop Route Request

The Node 4 matches the request packet destination with its name, after this match at node 4, it will initiate a Route Reply packet.

### Route Reply

The route reply packet format is given as:

**RREP: PacketDest: <previousNode>: <Node1>: <Node4>: <Node1 Node2 Node3 Node4>**

Node 4 will send the route reply back, according to the route obtained. The previous node name is extracted, and the reply is unicasted to that node. The previous node in this case is Node 3, which matches that either is the final destination node according to the route reply packet. Since it's not, it extracts the previous node from the route and forwards this route reply as it is to that node.

In this way, the reply packet finally reaches Node 1, which is the source node.

After the route is discovered at Node 1, the captureSensorData() function is invoked and a data packet containing GPS coordinates and Acceleration in the x y, and z axis as payload data is sent on the route discovered. This data packet is routed hop by hop to the route discovered.

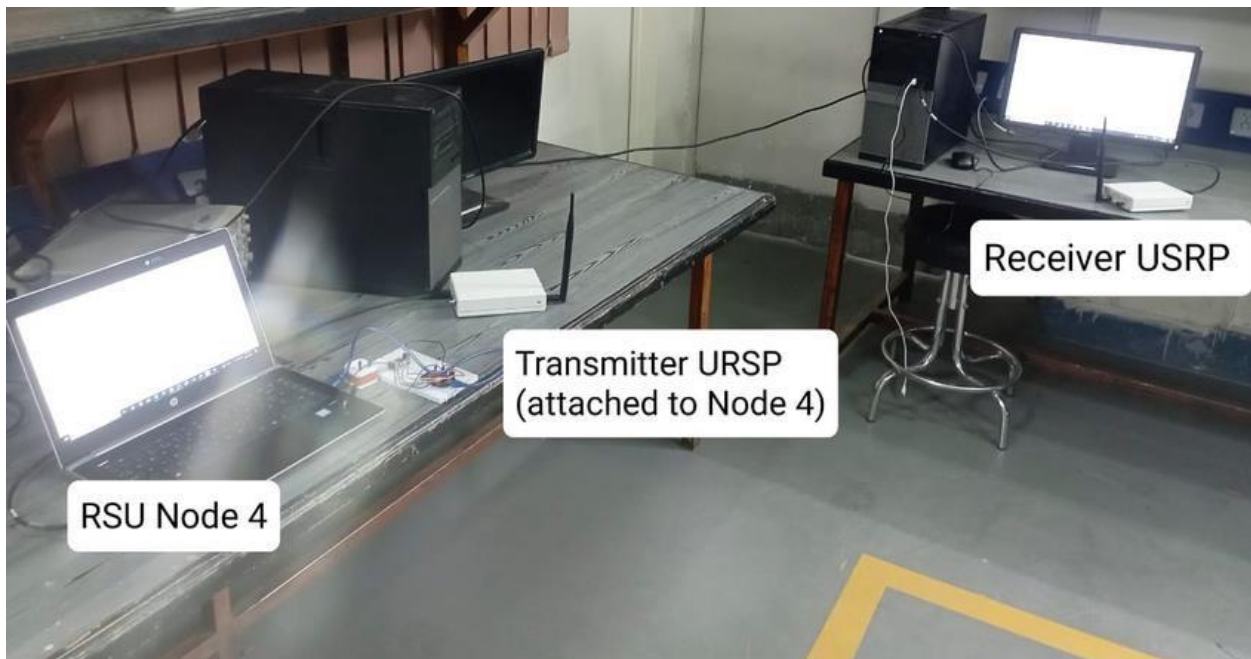


## Dynamic Behaviour

In DSR implementation, we have ensured dynamic topology by changing the location of nodes, as the nodes change their order the route discovered is changed accordingly. If a node in the route shuts down the request still reaches the destination node the route is adjusted accordingly, so the request still reaches the destination.

### 3.4 OFDM-Based USRP Communication

A USRP-based software-defined radio B-210 is connected to the RSU Node. The data collected at the RSU node is transferred to the GNU Radio software. This data is input to the OFDM transmitter, which modulates and sends it over the air. This OFDM signal is wirelessly received on another USRP, demodulation is performed and original data is recovered and stored in a file. The objective of this implementation is to demonstrate the OFDM transmission and reception which is the key technology of Wifi, 4G, and 5G systems.



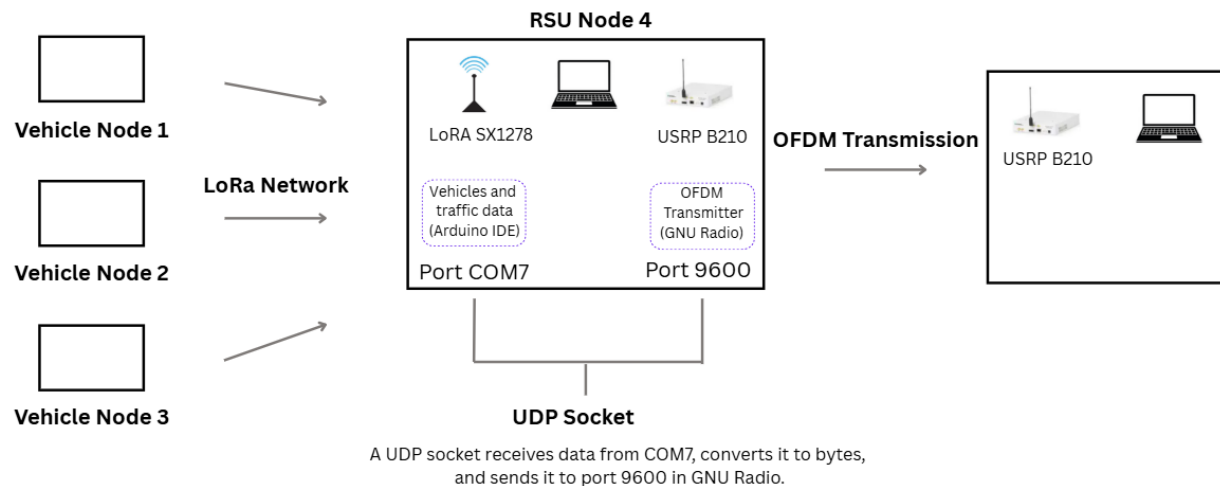
*Figure 8 Setup for OFDM Transmitter Receiver*

### 3.4.1 Connection between Arduino IDE and GNU Radio

RSU Node is displaying results on Arduino IDE software. These results are vehicular data, which are beacon messages, GPS coordinates, acceleration data received from the first three nodes, and traffic data extracted from a video file. This data is to be transmitted from USRP 1 using OFDM transmission, received by another URSP over the air, demodulated, and extracted back. For this, we need to transmit the incoming data from Arduino IDE to GNU radio, where the flow graph for the OFDM transmitter is designed.

The RSU Node is connected to the COM7 port of the laptop, where it receives the data. For this, we have made a UDP(User Datagram Protocol) socket in Python which reads data from Arduino IDE's Communication Port 7 and then sends it through a **UDP port (9600)**. This UDP Port address is defined in the UDP Source block in the transmitter flow graph. By running the Python script in the background, incoming data at Communication Port 7, is sent to UDP Port 9600, which is read by the UDP Source block and is fed to the OFDM Transmitter.

Flowgraph of OFDM Transmission and Reception



### 3.4.2 Transmitter Design

The OFDM transmitter is implemented using the GNU Radio Companion software, where various blocks are made and corresponding parameters for each block are defined. Following is the flow graph of the transmitter:

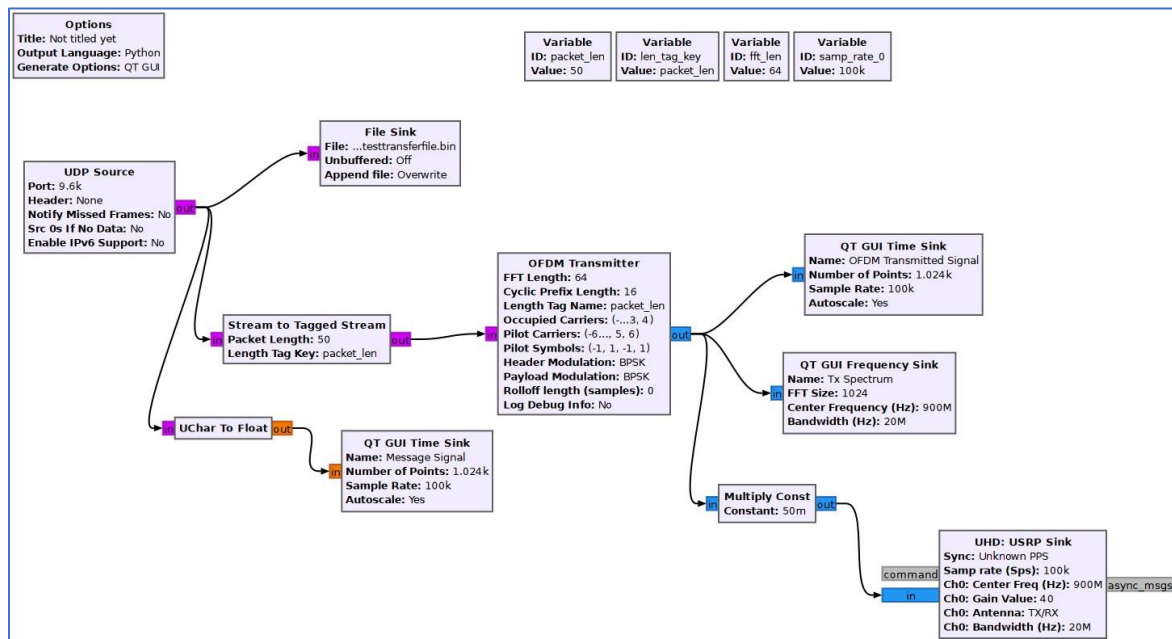


Figure 9 OFDM Transmitter

UDP Source is receiving data at port number 9600 in the form of ASCII characters, from a Python script running in the background. This data is converted into a stream of binary bits by **Stream to Tagged Stream block**. This block divides the incoming data into a fixed group size of 50 bytes set by the variable named packet\_len in the setting. It also adds a tag before this fix-sized packet, enabling the transmitter to identify the start of the packet.

Properties: Stream to Tagged Stream		
General    Advanced    Documentation		
Type	byte	
Vector Length	1	[int]
Packet Length	packet_len	[int]
Length Tag Key	len_tag_key	[string]

OK    Cancel    Apply

Figure 10 Stream to Tagged Stream block parameters

A carrier wave using the central frequency defined as 900MHz is created by this block. The incoming binary data is then mapped at the phases of carrier wave. The resulting waveform has a phase shift of **0 degrees for 1** and **180 degrees for 0**, which are then mapped onto subcarriers defined by occupied carriers(-4, -3, -2, -1, 1, 2, 3, 4) in the transmitter block's setting. This process places the modulated data onto the different frequencies of the frequency domain. Total Bandwidth which is set to be 20MHz, is divided into 64 subbands, with subcarrier spacing as:

$$\begin{aligned}
 \text{Subcarrier Spacing} &= \frac{\text{Bandwidth}}{\text{FFT length}} \\
 &= \frac{20\text{MHz}}{64} \\
 &= 312.5 \text{ kHz}
 \end{aligned}$$

### 3.4.3 Inverse Fast Fourier Transform

After the placement of data on the subcarriers, Inverse Fast Fourier Transform is applied to these subcarriers, which combines all the frequency components of subcarriers into a single time domain

signal that can be transmitted over the channel.

### 3.4.3.1 Cyclic Prefix

A cyclic prefix of 16 samples is added to the time domain signal to reduce **Inter-Symbol Interference** caused by multipath propagation. This cyclic prefix preserves our original data from interference and helps the receiver to identify the start of the OFDM signal. The receiver starts decoding after this cyclic prefix.

### 3.4.3.2 Pilot Carriers and Symbols

Pilot Carriers and the corresponding symbols are added, in order to enable the receiver to match these symbols at these specified locations. On the receiver side, pilot carriers at some frequencies (-6, -5, 5, 6) carry known predefined symbols (-1, 1, -1, 1) that are not data-modulated. The receiver estimates channels by correlating the received symbols with known symbols. On mismatch, the receiver identifies channel distortions such as frequency offsets or phase shifts. The mismatch information allows the receiver to estimate and correct channel degradations such as fading or interference and subsequently adjust the remaining data symbols to recover the data correctly.

Properties: OFDM Transmitter		
General   Advanced   Documentation		
FFT Length	fft_len	[int]
Cyclic Prefix Length	fft_len//4	[int]
Length Tag Name	len_tag_key	[string]
Occupied Carriers	((-4,-3,-2,-1,1,2,3,4),)	
Pilot Carriers	((-6,-5,5,6),)	
Pilot Symbols	((-1,1,-1,1),)	
Sync Word 1	None	
Sync Word 2	None	
Header Modulation	BPSK ▼	
Payload Modulation	BPSK ▼	
Rolloff length (samples)	0	[int]
Scramble Bits	No ▼	
Log Debug Info	No ▼	
		OK   Cancel   Apply

Figure 11 Transmitter block parameters

The OFDM transmitter's output is connected to the USRP Source block which transmits the time domain OFDM signal over the air.

Parameter	Value
Central Frequency	900 MHz
Gain	40
Bandwidth (BW)	20 MHz
Sample Rate	100 kHz

*Table 6 USRP RF settings*

#### **3.4.4 Receiver Design**

The same subcarriers, cyclic prefixes, and pilot carrier settings are made on the receiver side. Also, the USRP block parameters are the same as the transmitter block.

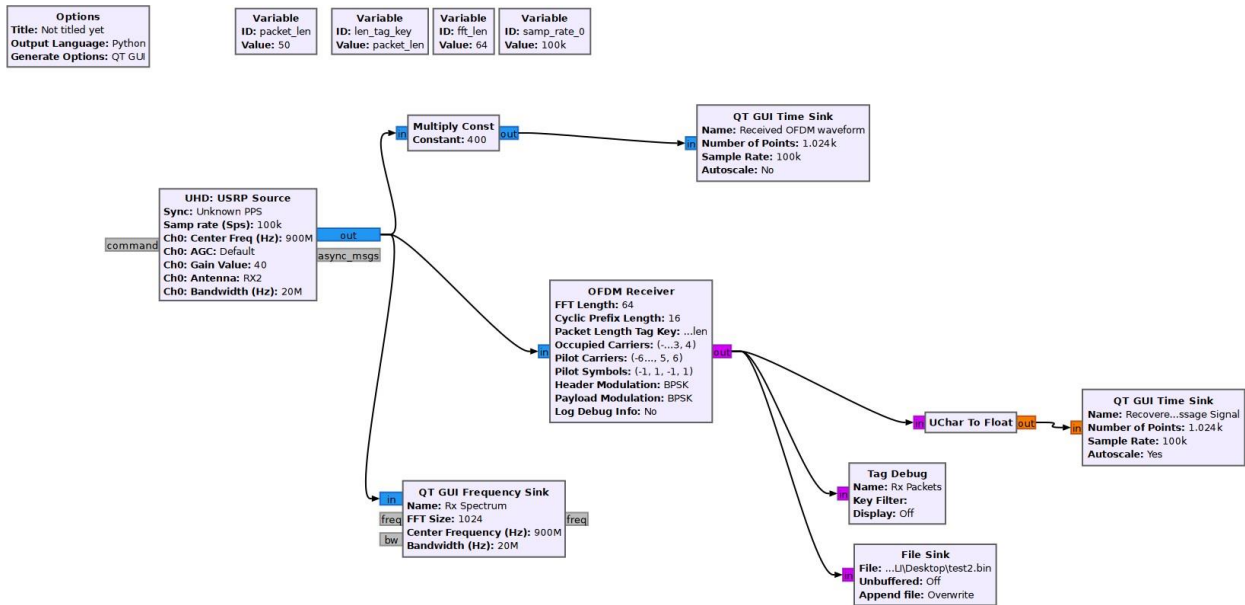


Figure 12 OFDM Receiver

The procedure begins by receiving the signal using the USRP Source, the signal is sampled and visualized by a Time Sink GUI graph block. The cyclic prefix is removed, and FFT is applied in order to extract the frequency domain subcarriers. These subcarriers are demodulated on the basis of phase shifts using BPSK. Finally, the original data bits are recovered and converted into readable text by the transmitter block. The output of the receiver is connected to a file sink block, in which an empty bin file path is provided, to store the recovered data.

Properties: OFDM Receiver	
General	Advanced
FFT Length	fft_len [int]
Cyclic Prefix Length	fft_len//4 [int]
Packet Length Tag Key	packet_len [string]
Occupied Carriers	((-4,-3,-2,-1,1,2,3,4),)
Pilot Carriers	((-6,-5,5,6),)
Pilot Symbols	((-1,1,-1,1),)
Sync Word 1	None
Sync Word 2	None
Header Modulation	BPSK
Payload Modulation	BPSK
Scramble Bits	No
Log Debug Info	No
<div>OK Cancel Apply</div>	

Figure 13 Receiver block parameters

### Vehicle-to-Infrastructure (V2I) :

This project implements a **Vehicle-to-Infrastructure (V2I)** framework to communicate congestion data between roadside infrastructure and vehicles. The workflow involves:

## V2I Workflow Diagram

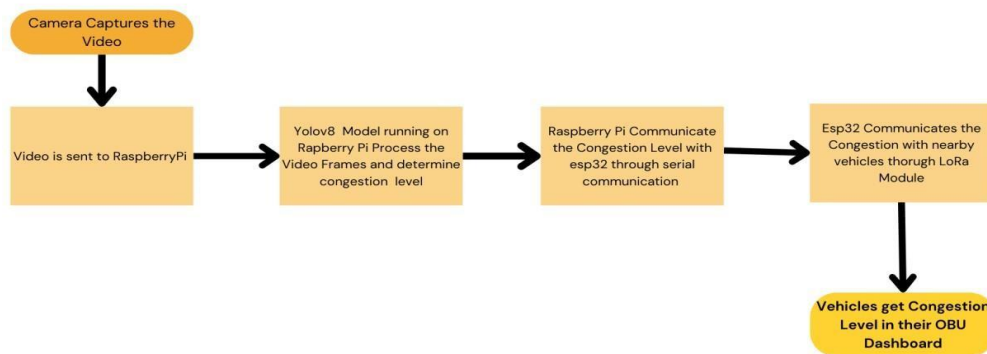


Figure 14 V2I Block Diagram

### 3.4.1 Vehicle Detection & Counting

#### ○ YOLOv8-Based Detection

- Cameras pick up vehicle movement on the streets in real time using YOLOv8 vehicle detection.
- **Class Filtering:** Using COCO dataset class IDs **2, 5, 7**, we only filter out vehicles — which are **cars, trucks, and motorcycles**.
- **Confidence Thresholding:** In order to enhance detection accuracy, scores below a certain threshold — in this case **0.5** — are removed.

#### ○ Region of Interest (ROI) Definition

- An example of an ROI that can be used for zooming purposes is **lane or intersection polygons** or defined boxes as areas of the street that need special attention.
- To ensure spatial correctness, the ROI position is **manually adjusted** through the camera perspective as the calibration for the camera's position.



### ○ Centroid Calculation

- The centroid of every vehicle bounding box is determined from the **geometrical center**.

$$\text{Centroid formula: } (\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}).$$

### ○ ROI-Based Counting

- The centroid position is determined to be within the useful area through the use of **OpenCV's pointPolygonTest ()** function.
- Maintaining an account of the counted vehicles allows controlling **double counting** of the vehicle using **vehicle tracking IDs**.
  - Congestion Level Determination:
    - Vehicle counts within the ROI over a fixed time window (e.g., 30 seconds) are categorized into:
      - **Low Congestion**: < 10 vehicles.
      - **Medium Congestion**: 10–20 vehicles.
      - **High Congestion**: > 20 vehicles.
    - Periodic resets of the count and tracking set adapt to dynamic traffic conditions.

## 3.4.2 Activities on Raspberry Pi Edge Processing

### ○ Setup

- A **Raspberry Pi 3B with 4GB RAM** is powerful enough to perform video decoding, model inference, and data transmission.
- **Traffic videos** can be captured using a **camera module** or **USB webcams**.
- **Congestion data** is transmitted to the Pi via **GPIO pins** or **USB ports**. The Pi then sends it to the **ESP32**.
- The system is cooled with **heat sinks** and **cooling fans** to manage thermal overload during prolonged use.

## ○ Model and Software Implementation

- **Memory and CPU usage** are optimized with the **64-bit Raspberry Pi OS**.
- The equipment is faster with **less accurate models**. Thus, the **YOLOv8 Nano (YOLOv8n)** model is deployed to retain a measure of accuracy.
- Optimizations include:
  - Lowering the precision to **16-bits**
  - Increasing input resolution
  - Setting boundaries of processing power (**Pi-capped frameworks**) for the Pi's calculations to compute within



*Figure 15 Raspberry pi Hardware*

## ○ Coupling with ESP32

- The **congestion levels (0, 1, 2)** are sent from the Raspberry Pi towards the ESP32 unit through **UART serial communication**.
- The data is sent in **compact forms** to enable **low transmission delay**.

### 3.5.3 From Communication through ESP32 to LoRa

- The **congestion data** is first transmitted from the **Raspberry Pi** to an **ESP32 controller** through **serial/UART**.
- The data is transmitted **wirelessly** with the aid of **LoRa (Long Range)**, which provides long-distance (>10 km), low power communication.
- **Changing vehicles** receive LoRa signals that give the **congestion level**, allowing **dynamic rerouting**.

### 3.5.4 Advantages of Using V2I Technology

- Enhanced Livestream Traffic Control  
Live congestion increases effortlessly through clefts, grenade streams.

- Economical & Expandable

The **ESP32** and **Raspberry Pi** form one of the **cheapest edge-computing devices** that can theoretically be placed on a **huge number of traffic lights**.

- Smart Streaming Sense and Efficiency
  - The **low power use** of each Pi is under **5 watts**, with autonomy guaranteeing the feeding of advanced LoRa systems.
  - Ideal for use with **solar-powered plants** at remote places.
- Edge-AI Integration

The **absence of cloud dependency** for localized processing on the Pi helps mitigate issues with **latency** and **bandwidth consumption**.

## **CHAPTER 4-RESULTS**

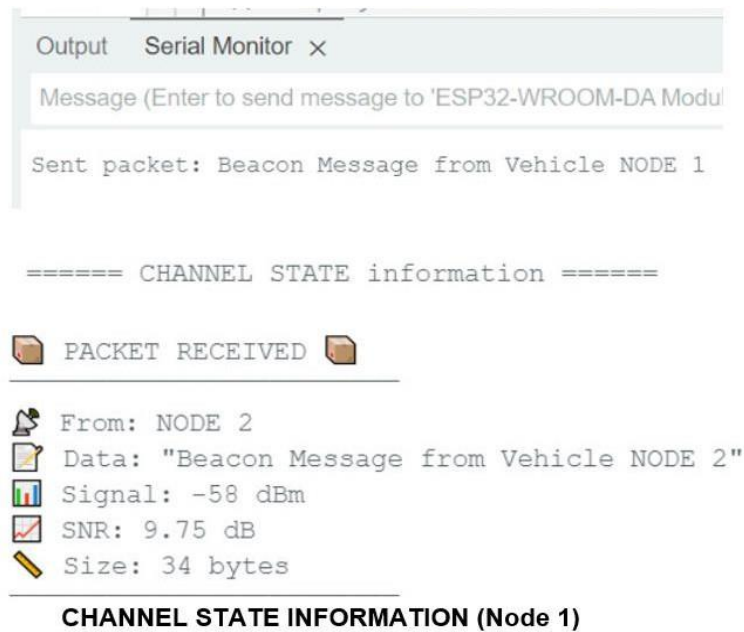
### **4.1 Direct Communication Mode**

In this section, the results of the **Direct Communication Mode** are discussed. In this mode, all nodes share data with each other directly without any intermediate routing. A node accurately identifies and displays data received from other nodes in real-time in addition to sending its own data. In the following each node is connected to a laptop, displaying results on Arduino IDE software.

#### **4.1.1 Channel State Information**

- **Channel between Node 1 and other Nodes:**

The beacon message broadcast from this node is displayed at the top. Afterward, the channel state information of a packet received from a nearby Node 2 is displayed. Signal-to-noise ratio, link quality using the received signal's strength(Signal written in image below) , and data parameters are processed and shown. The received signal strength (RSSI) of Node 1 from Node 2 is -58 dBm, which is a strong and stable wireless link. A greater (less negative) RSSI value indicates little signal loss through the channel, guaranteeing successful communication. This reading implies Node 2 is near or within a good transmission range greater (less negative) RSSI value indicates little signal loss through the channel, guaranteeing successful communication. This reading implies Node 2 is near or within a good transmission range



*Figure 16 Channel state information (node 1)*

In the similar way packets from Node 3 and Node 4 have been received and processed by this node.

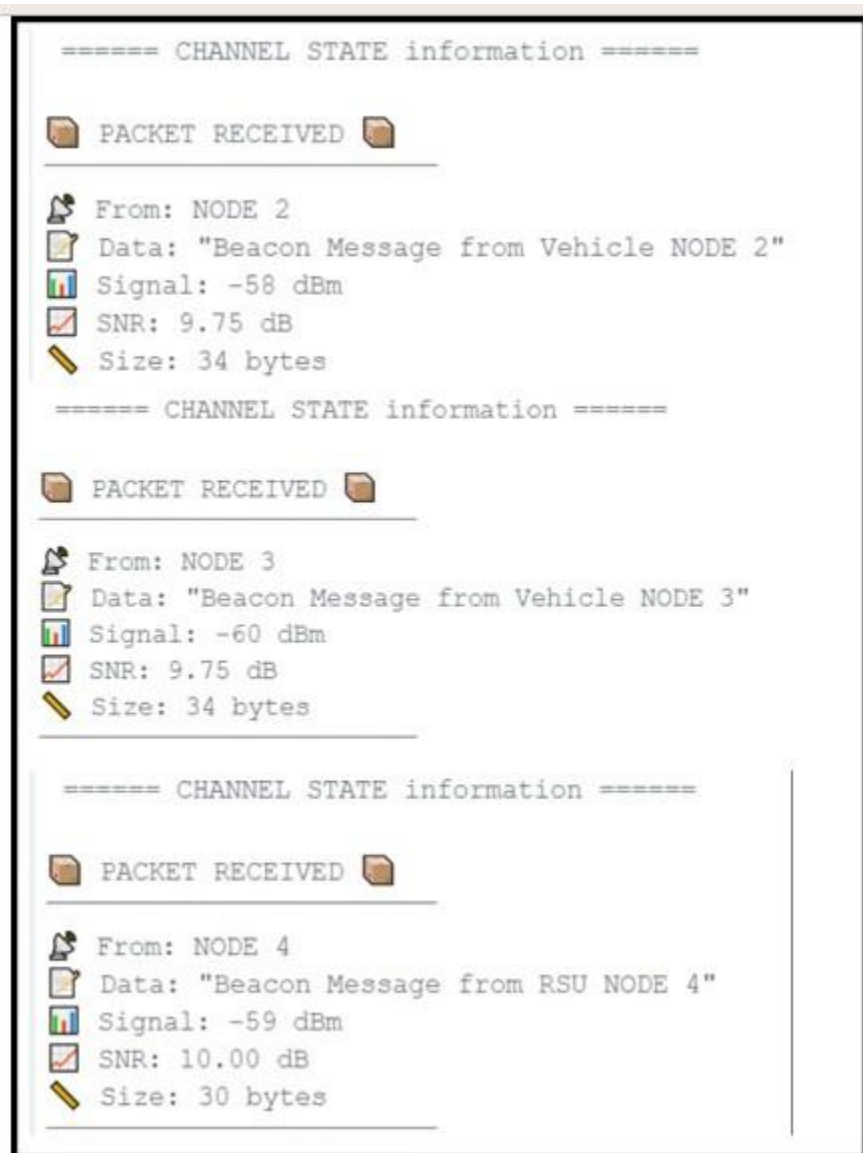


Figure 17 Channel state information (node 2, node 3, node 4)

- **Channel between Node 2 and other Nodes:**

Similarly at Node 2, the packets are being received, processed, and accurately displayed in real-time for network assessment. Also the packet broadcasted by it is displayed as well.

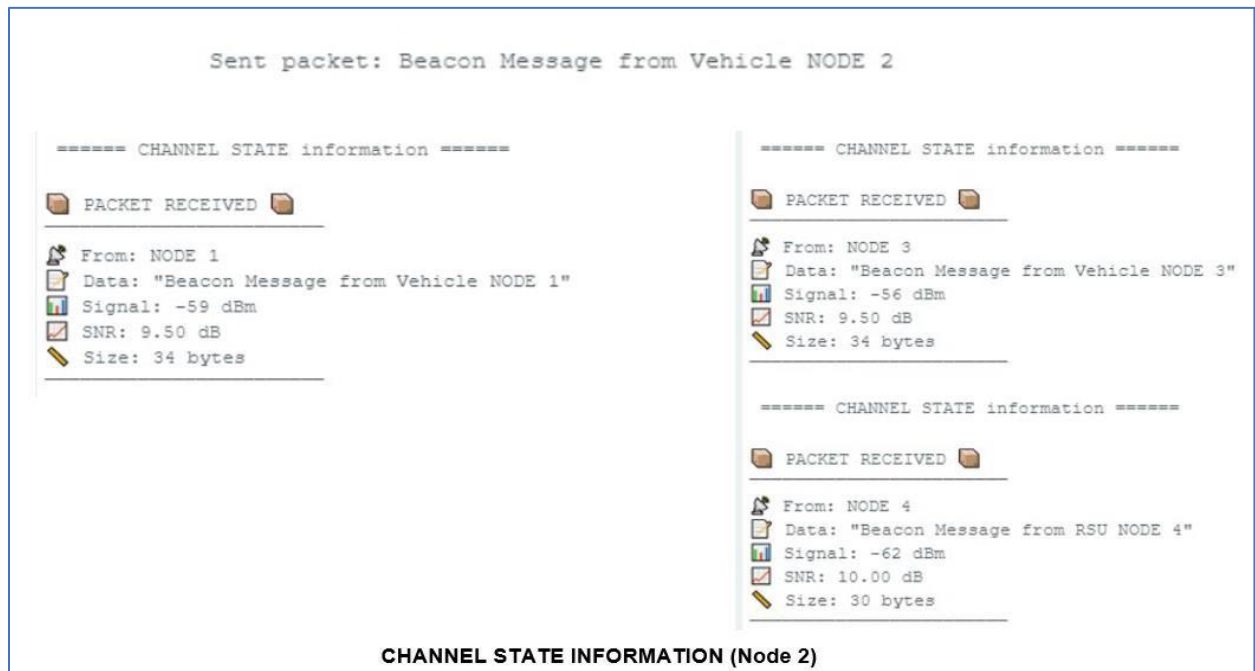


Figure 18 Channel state information( node 2)

- **Channel State Information of Node 3 and Node 4:**

The channel state information of Node 3 and Node 4 are displayed respectively:

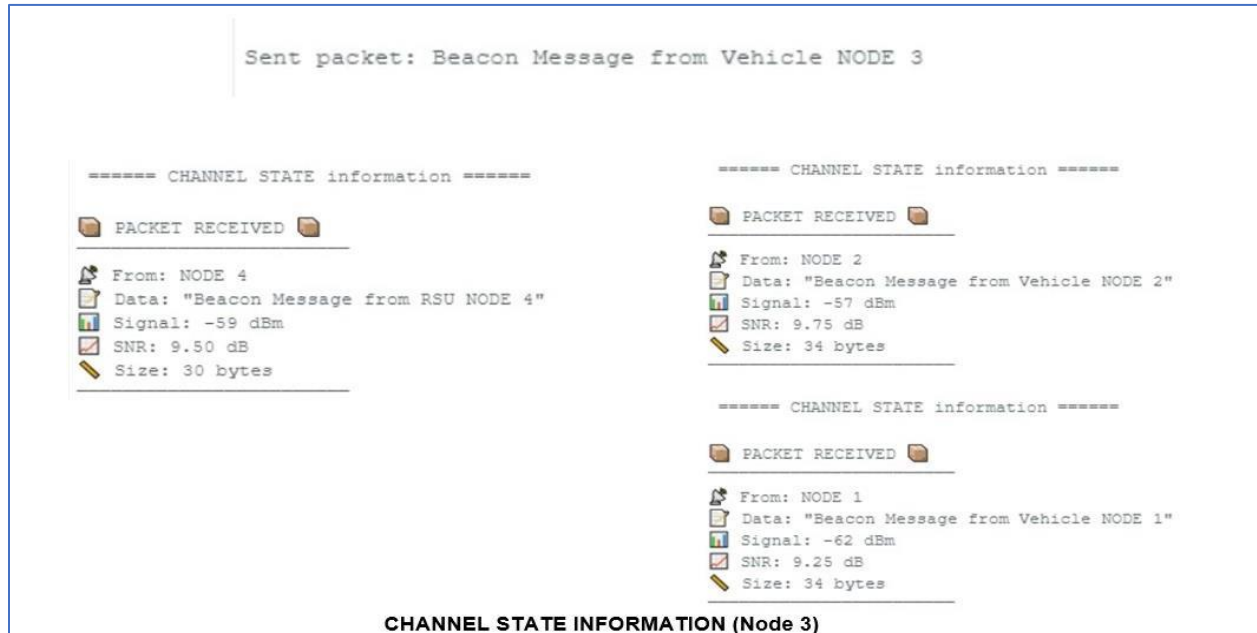


Figure 19 Channel State information (node 3)

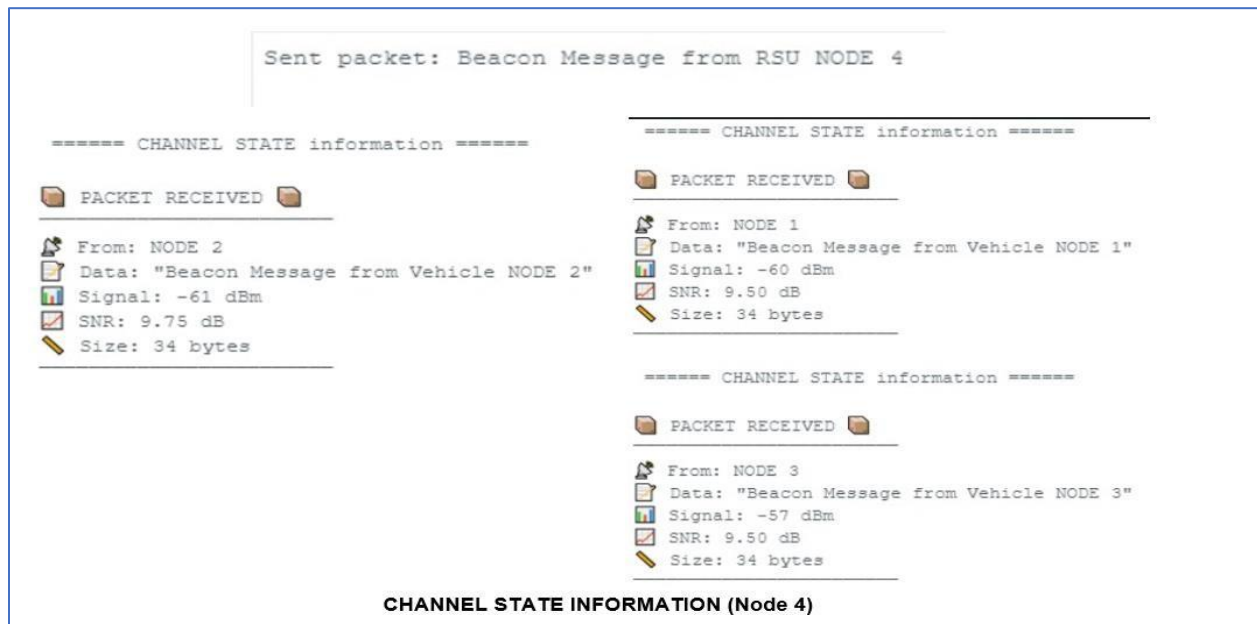


Figure 20 Channel State information(node 4)

#### 4.1.2 Node State Information



- **Node 1**

In node state information, the distance of a node from which a packet is received is calculated and displayed. Last data received and signal strength are also displayed. From this distance, nearest node is determined and shown.

```
===== Current NODE STATE Information =====
Current Node: NODE 1
📶 Nearest Node: NODE 2 (Distance: 7.1m)

Data recieved from vehicles:

From: "NODE 3"
Last Data: "Beacon Message from Vehicle NODE 3"
Signal Strength: -60 dBm
Total Packets: 4

-----

From: "NODE 4"
Last Data: "Beacon Message from RSU NODE 4"
Signal Strength: -60 dBm
Total Packets: 4

-----

From: "NODE 2"
Last Data: "Beacon Message from Vehicle NODE 2"
Signal Strength: -57 dBm
Total Packets: 4

-----

Sent packet: Beacon Message from Vehicle NODE 1
```

*Figure 21 Node state information (node 1)*

- **Node 2:**

Similarly the Node state information of Node 2 has been displayed. Nearest node, signal strength, and data received from other nodes is extracted from the received packets and shown.

```
===== Current NODE STATE Information =====
```

```
Current Node: NODE 2
```

```
📶 Nearest Node: NODE 3 (Distance: 5.6m)
```

```
Data recieved from vehicles:
```

```
From: "NODE 1"
```

```
Last Data: "Beacon Message from Vehicle NODE 1"
```

```
Signal Strength: -61 dBm
```

```
Total Packets: 5
```

```
From: "NODE 4"
```

```
Last Data: "Beacon Message from RSU NODE 4"
```

```
Signal Strength: -60 dBm
```

```
Total Packets: 6
```

```
From: "NODE 3"
```

```
Last Data: "Beacon Message from Vehicle NODE 3"
```

```
Signal Strength: -55 dBm
```

```
Total Packets: 5
```

```
Sent packet: Beacon Message from Vehicle NODE 2
```

### **NODE STATE INFORMATION (Node 2)**

*Figure 22 Node state information (node 2)*

- **Node 3 and Node 4:**

```

===== Current NODE STATE Information =====
Current Node: NODE 3
📶 Nearest Node: NODE 2 (Distance: 6.3m)

Data recieved from vehicles:

From: "NODE 4"
Last Data: "Beacon Message from RSU NODE 4"
Signal Strength: -58 dBm
Total Packets: 5

From: "NODE 2"
Last Data: "Beacon Message from Vehicle NODE 2"
Signal Strength: -56 dBm
Total Packets: 5

From: "NODE 1"
Last Data: "Beacon Message from Vehicle NODE 1"
Signal Strength: -62 dBm
Total Packets: 5

Sent packet: Beacon Message from Vehicle NODE 3

```

**NODE 3**

*Figure 23 Node state information (node 3)*

```

===== Current NODE STATE Information =====
Current Node: NODE 4
📶 Nearest Node: NODE 3 (Distance: 7.1m)

Data recieved from vehicles:

From: "NODE 2"
Last Data: "Beacon Message from Vehicle NODE 2"
Signal Strength: -59 dBm
Total Packets: 6
-

From: "NODE 3"
Last Data: "Beacon Message from Vehicle NODE 3"
Signal Strength: -57 dBm
Total Packets: 7
-

From: "NODE 1"
Last Data: "Beacon Message from Vehicle NODE 1"
Signal Strength: -58 dBm
Total Packets: 7
-

Sent packet: Beacon Message from RSU NODE 4

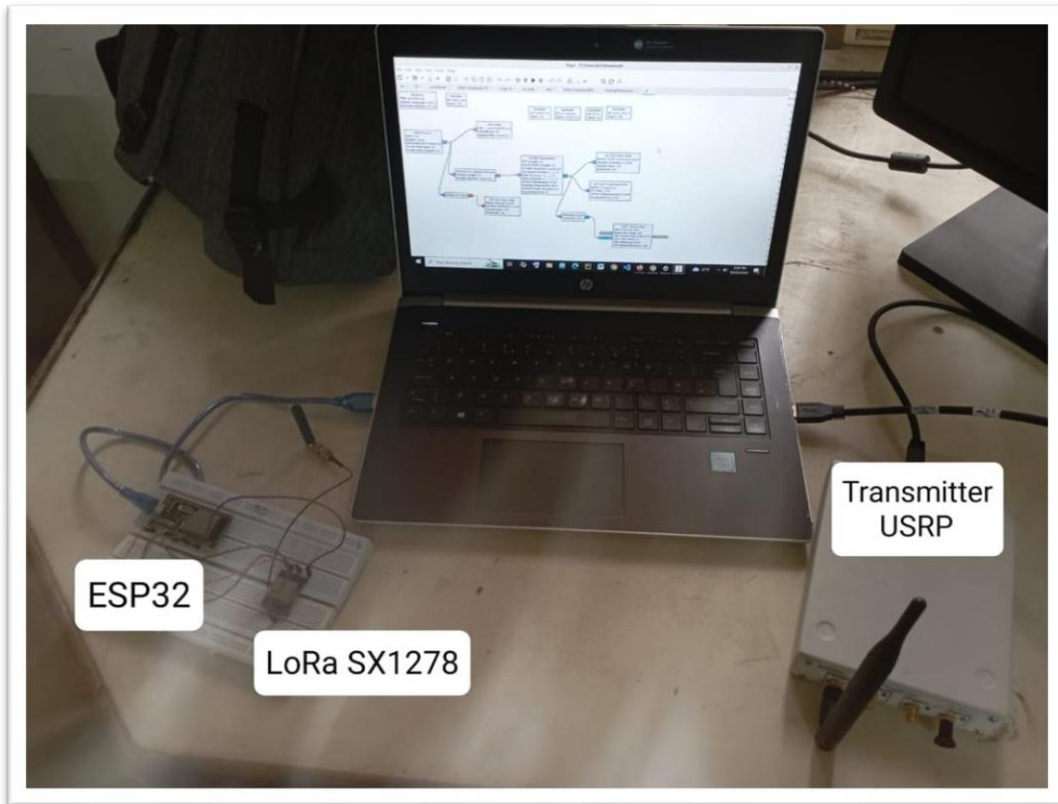
```

**NODE 4**

*Figure 24 Node state information (node 4)*

### 4.1.3 Road Side Unit prototype Node 4

A USRP-B210 has been attached with the same laptop with Node 4 connected:



*Figure 25 Setup for Node 4 with USRP attached*

For this we have placed the three nodes at close distance in order for them to send their respective sensor data (GPS coordinates and Acceleration). Since these nodes are in close proximity of Node 4, the sensors data at the vehicle nodes will be fetched and send instead of a beacon message.

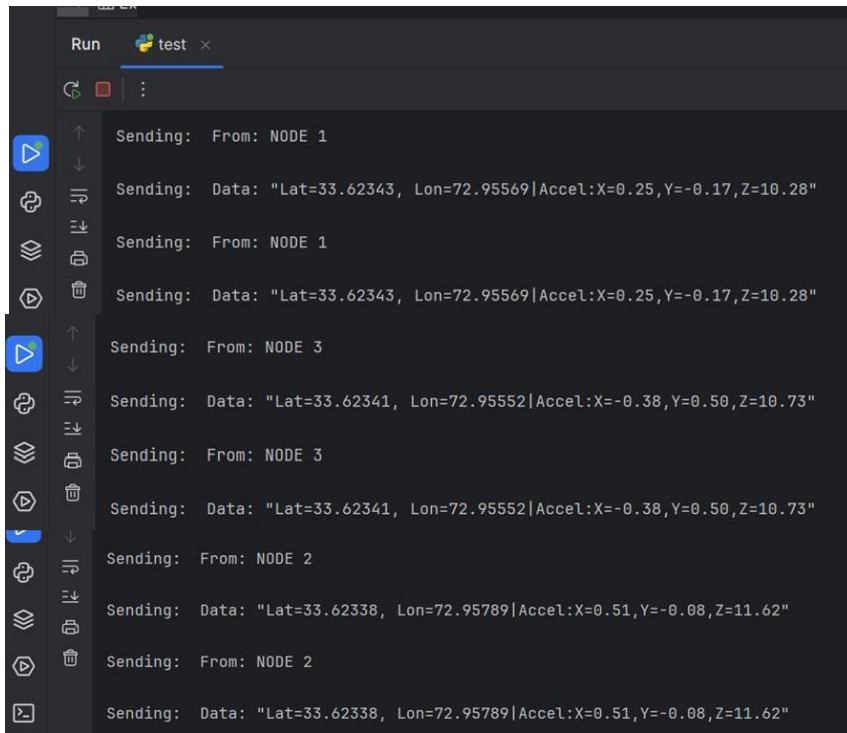
A script of UDP Socket running on the laptop takes data from COM Port 7 to Port 9600. This 9600 port address is defined in the UDP Source block of OFDM transmittter block in GNU Radio.

### 4.1.3 Node 4 Data Transmission and Reception via OFDM

#### 4.1.3.1 Data Transmission from Nodes 1, 2, and 3 (TX Side)

- **Data Transferred from Node 1, 2, 3 to Node 4 via Python Script:**

Following are the snapshots of the vehicle nodes data transfer to Port 9600 from a python script:



```
Run test x
Sending: From: NODE 1
Sending: Data: "Lat=33.62343, Lon=72.95569|Accel:X=0.25,Y=-0.17,Z=10.28"
Sending: From: NODE 1
Sending: Data: "Lat=33.62343, Lon=72.95569|Accel:X=0.25,Y=-0.17,Z=10.28"
Sending: From: NODE 3
Sending: Data: "Lat=33.62341, Lon=72.95552|Accel:X=-0.38,Y=0.50,Z=10.73"
Sending: From: NODE 3
Sending: Data: "Lat=33.62341, Lon=72.95552|Accel:X=-0.38,Y=0.50,Z=10.73"
Sending: From: NODE 2
Sending: Data: "Lat=33.62338, Lon=72.95789|Accel:X=0.51,Y=-0.08,Z=11.62"
Sending: From: NODE 2
Sending: Data: "Lat=33.62338, Lon=72.95789|Accel:X=0.51,Y=-0.08,Z=11.62"
```

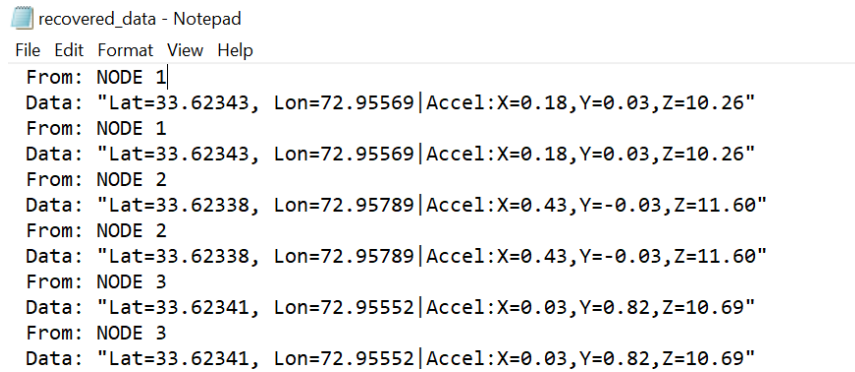
*Figure 26 Data Transferred from Node 1, 2, 3 to Node 4 via Python Script*

This GPS and Acceleration data of vehicles nodes are received by the UDP Source block in GNU Radio. From there, this data is fed into the OFDM Transmitter block, modulated and transmitted over the air from the USRP Radio.

#### **4.1.3.2 Data Reception and Recovery on Receiver side USRP**

Another USRP B210 attached with a laptop receives this ofdm modulated data, demodulates, extracts and stored the original vehicle nodes data into a bin file. The Receiver flow graph is running on the GNU Radio. In this graph after the OFDM block's output is connected to a File Sink block, in which path for an empty file named as recovered\_data is provided. This purpose of this is to store the recovered data and verify the results.

- **Recovered Data Stored in Binary File:**



```
recovered_data - Notepad
File Edit Format View Help
From: NODE 1
Data: "Lat=33.62343, Lon=72.95569|Accel:X=0.18,Y=0.03,Z=10.26"
From: NODE 1
Data: "Lat=33.62343, Lon=72.95569|Accel:X=0.18,Y=0.03,Z=10.26"
From: NODE 2
Data: "Lat=33.62338, Lon=72.95789|Accel:X=0.43,Y=-0.03,Z=11.60"
From: NODE 2
Data: "Lat=33.62338, Lon=72.95789|Accel:X=0.43,Y=-0.03,Z=11.60"
From: NODE 3
Data: "Lat=33.62341, Lon=72.95552|Accel:X=0.03,Y=0.82,Z=10.69"
From: NODE 3
Data: "Lat=33.62341, Lon=72.95552|Accel:X=0.03,Y=0.82,Z=10.69"
```

*Figure 27 Recovered Data stored in a file*

The data is accurately recovered by OFDM receiver and is displayed in the above figure.

## 4.2 SNR, RSSI AND DISTANCE GRAPHS

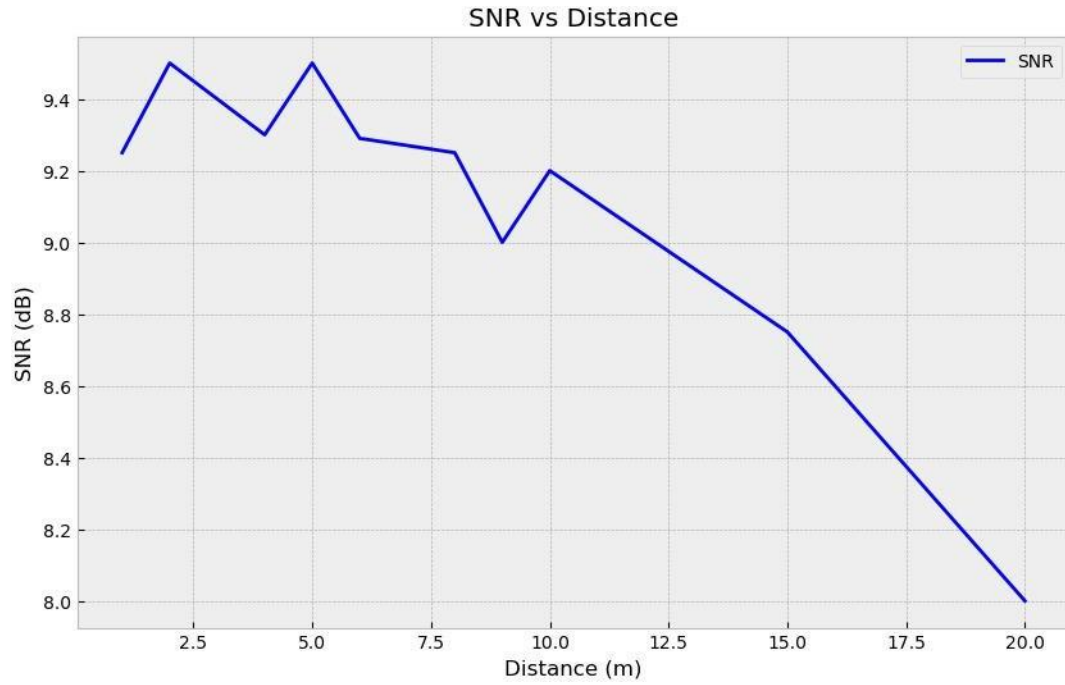


Figure 28 SNR Vs Distance graph

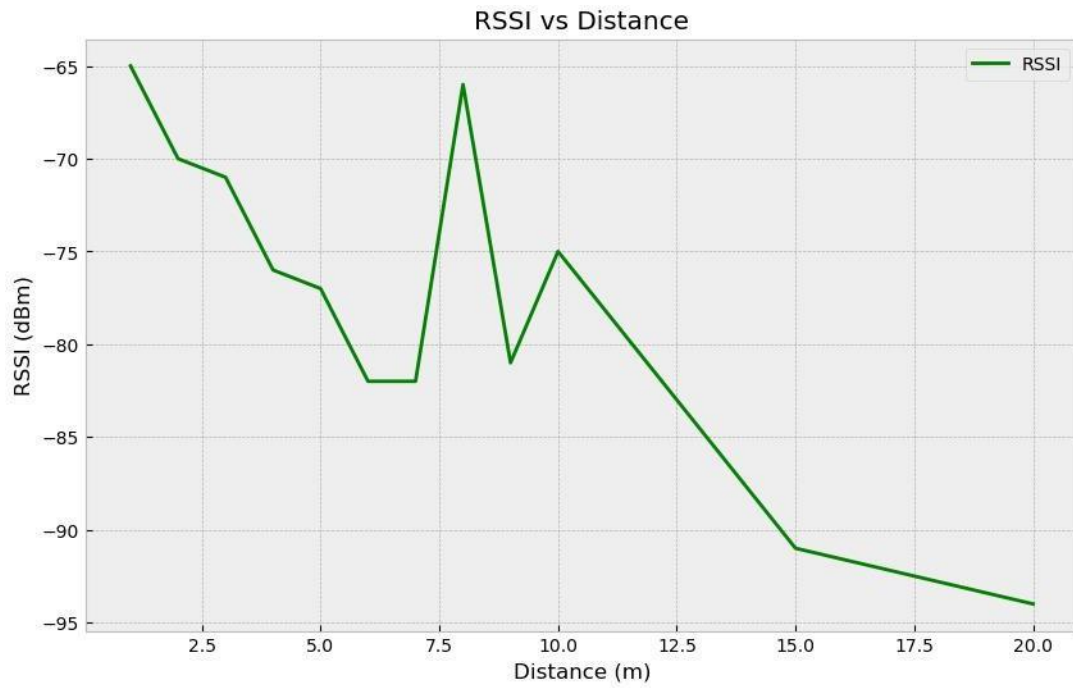


Figure 29 RSSI Vs Distance graph

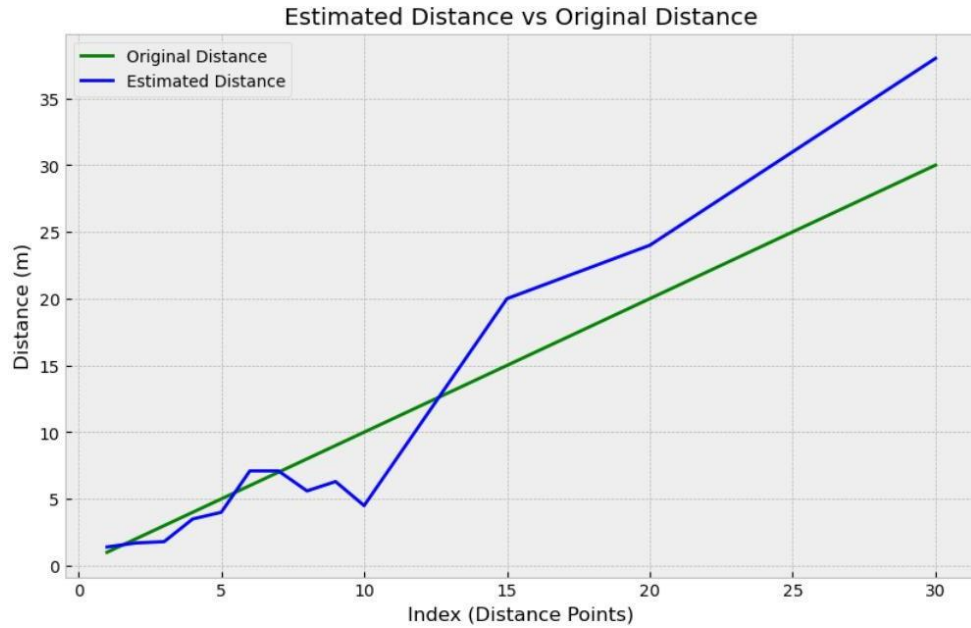


Figure 30 Estimated Distance Vs Actual Distance graph

### 4.3 Time and Frequency Spectrum Analysis

- **Transmitter Side Spectrum:**

Following are the Time and Frequency domain plots of the transmitter side:

The below graph shows the time domain OFDM signal transmitted over the air. The red and blue graphs are the In-phase and Quadrature components, respectively. These are results of BPSK Modulation on the subcarriers. Original data lies in the In-phase component since the Quadrature component vanishes because of 0 and 180 degrees of phase shift.



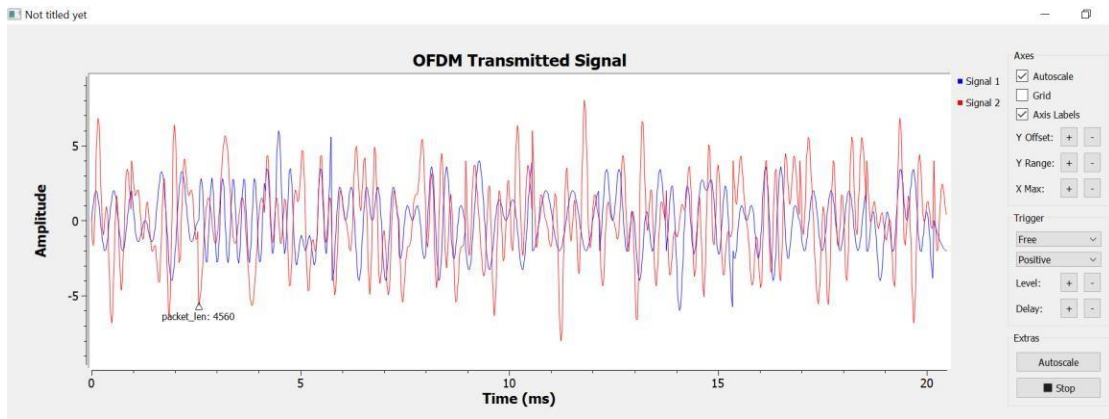


Figure 31 OFDM Transmitted Signal

Below is the graph of Frequency spectrum of OFDM Transmitter, centered around 900MHz over a bandwidth of 20MHz (from **890 MHz to 910 MHz**). The peaks in the signal shows the modulated subcarriers containing the data at specific frequencies.

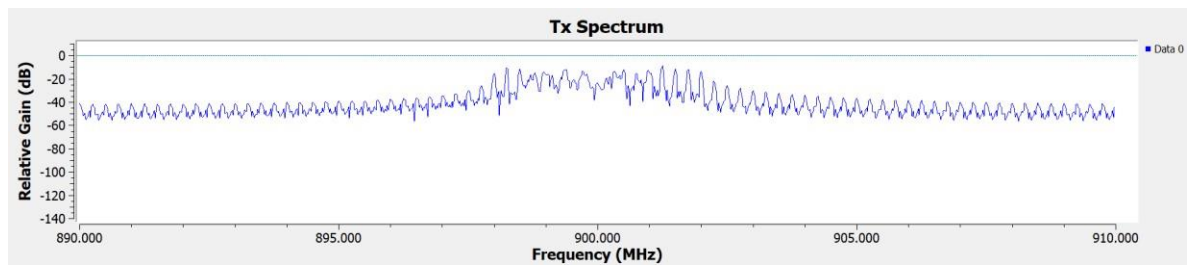


Figure 32 Tx Spectrum

This plot shows the time domain signal of the actual data being transmitted. The variation in the blue graph shows how amplitude of the data changes over time. The sharp peaks and oscillations show that data is transmitted in discrete chunks.

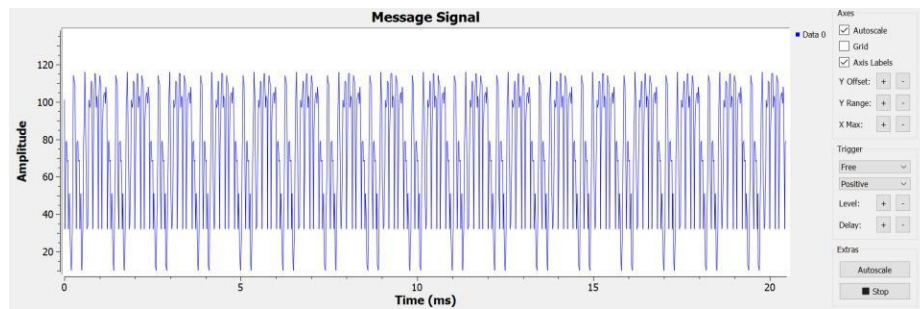


Figure 33 Message Signal

- **Receiver Side Spectrum:**

Following are the receiver side Time and Frequency domain spectrums.

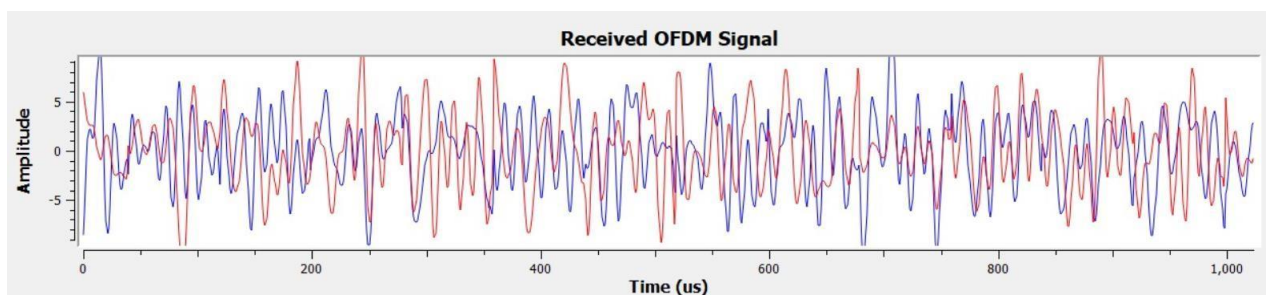


Figure 34 Received OFDM Signal

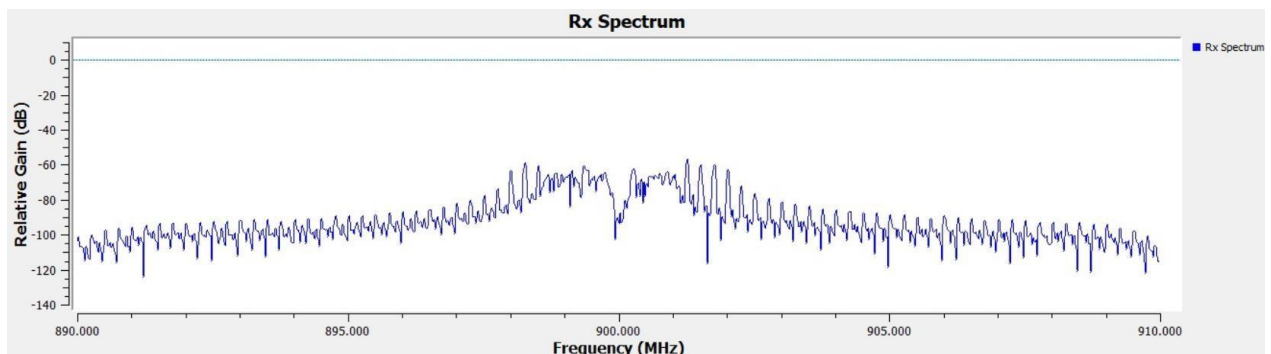


Figure 35 Rx Spectrum

## Recovered Message Signal Plot

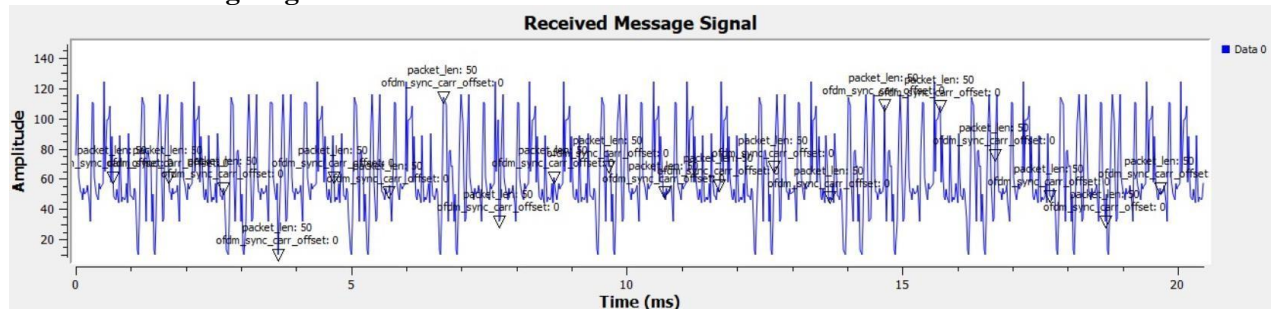


Figure 36 Received Message Signal

The graph shows the amplitude of the recovered data and is similar to the one transmitted from Node 4.

## 4.4 Dynamic Source Routing Protocol

Following is the setup of nodes, placed apart with each connected to a laptop displaying results on Arduino IDE software.



Figure 37 Nodes setup for implementing DSR

### 4.4.1 Route Discovery

Source Node 1 initiates a beacon in order to determine a nearest node, to find route to reach the destination node. Node 2 is the nearest node, which receives this packet and generates a beacon reply message.

```
Received Beacon Reply from: Node2 | RSSI: -78
Received Another Beacon Reply from: Node2 | RSSI: -78
Received Another Beacon Reply from: Node2 | RSSI: -77
📌 Nearest Node Selected: Node2
🚀 ROUTE REQUEST SENT 🚀
Source: Node1
Destination: Node4

✅ ACK Received from Node2
```

Figure 38 link established between two nodes

A link is established between the two nodes from thi beacon reply.

```

BEACON sent back successfully
-----
Preparing to send back Beacon Reply
BEACON sent back successfully
-----
Waiting for the packets

```

Figure 39 Node 2 matches the destination id

Node 2 matches the destination id name with its own id, since its not the destination so it finds its next nearby node to forward the route request . A Beacon Reply at this node , from nearby Node 3 is received.

```

Received Another Beacon Reply from: Node3 | RSSI: -86
Received Another Beacon Reply from: Node3 | RSSI: -90
Received Another Beacon Reply from: Node3 | RSSI: -90
📌 Nearest Node Selected: Node3
Forwarded Route Request

```

Figure 40 Route Request is forwarded to Node 3

Route Request is forwarded to Node 3 , since its not the destination node as well, so in the similar way finds its next nearby node, and forwards this received request.

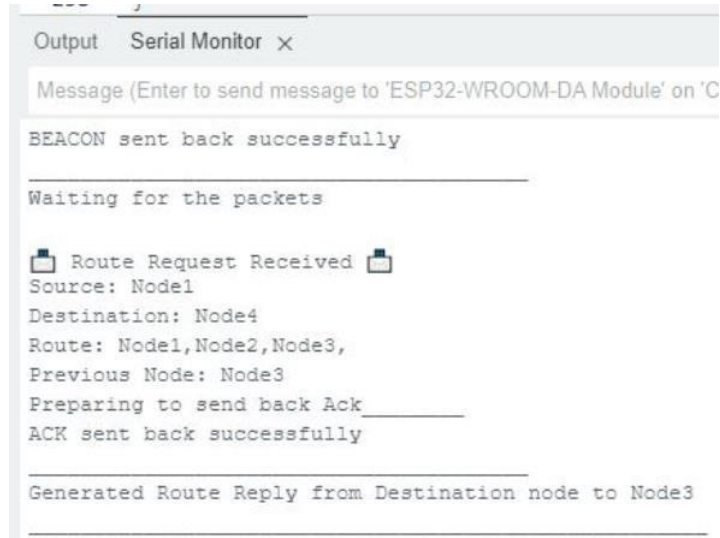
```

📧 Route Request Received 📧
Source: Node1
Destination: Node4
Route: Node1,Node2,
-----
BEACON msg sent successfully
Received Beacon Reply from: Node4 | RSSI: -85
Received Another Beacon Reply from: Node4 | RSSI: -85
Received Another Beacon Reply from: Node4 | RSSI: -86
Received Another Beacon Reply from: Node4 | RSSI: -87
📌 Nearest Node Selected: Node4
Forwarded Route Request

```

Figure 41 request arrives from multihop at Node 4

Finally the route request arrives from multihop at Node 4.



```
Output  Serial Monitor x
Message (Enter to send message to 'ESP32-WROOM-DA Module' on 'C
BEACON sent back successfully

Waiting for the packets

📧 Route Request Received 📧
Source: Node1
Destination: Node4
Route: Node1,Node2,Node3,
Previous Node: Node3
Preparing to send back Ack
ACK sent back successfully

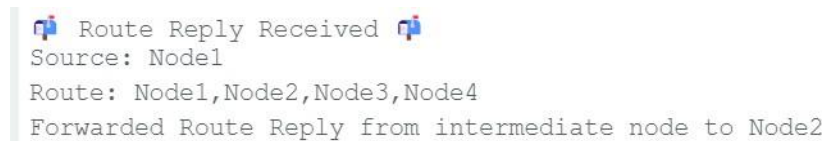
Generated Route Reply from Destination node to Node3
```

Figure 42 At Node 4, the destination id will be matched

At Node 4, the destination id will be matched with the Node name, so it generates a RouteReply on the same path in the reverse order in which request is received.

#### 4.4.2 Route Reply

The destination Node 4 unicasts the route reply packet by appending its own name in the route towards the Node 3. This reply is forwarded by Node 2 and Node 3 and finally route reply packet arrives at Source Node



```
📧 Route Reply Received 📧
Source: Node1
Route: Node1,Node2,Node3,Node4
Forwarded Route Reply from intermediate node to Node2
```

Figure 43 Node 3 forwarding route reply

1.

```
📡 Route Reply Received 📡  
Source: Node1  
Route: Node1,Node2,Node3,Node4  
Generating Route Reply  
  
Forwarded Route Reply from intermediate node to Node1
```

*Figure 44 Node 2 forwarding route reply*

```
📡 Route Reply Received 📡  
Source: Node1  
Route: Node1,Node2,Node3,Node4  
  
🔍 ROUTE DISCOVERED 🔍  
Full Route: Node1,Node2,Node3,Node4  
Waiting for route replies
```

*Figure 45 Route Reply arrived at Node 1*

Route Reply arrived at Node 1. After receiving this packet, Source node fetches the sensor data which are GPS longitudes latitudes, and Acceleration in x,y and z axis, prepares a data packet and forwards this on the discovered route hop by hop. Objective is to route this data successfully to the destination Node 4.

```
📡 Data Packet Sent: DATA:PacketDest:Node2:ROUTE:Node1,Node2,Node3,Node4:SOURCE:Node1:DESTINATION:Node4:PAYLOAD:33.622923,72.958043  
Waiting for route replies  
✅ Valid GPS Data Captured: 33.624214,72.957679  
Next node is: Node2  
📡 Data Packet Sent: DATA:PacketDest:Node2:ROUTE:Node1,Node2,Node3,Node4:SOURCE:Node1:DESTINATION:Node4:PAYLOAD:33.624214,72.957679  
Waiting for route replies  
✅ Valid GPS Data Captured: 33.625242,72.957944  
Next node is: Node2
```

*Figure 46 Data packet containing the sensor data*

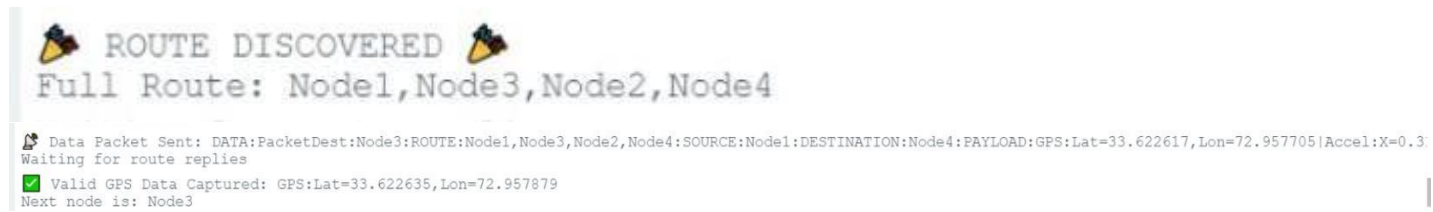
This data packet containing the sensor data reaches successfully at the Node 4.

```
Data Packet Received successfully as follows: ✅  
  
DATA:PacketDest:Node4:ROUTE:Node1,Node2,Node3,Node4:SOURCE:Node1:DESTINATION:Node4:PAYLOAD:GPS:Lat=33.622677, Lon=72.957696|Accel:X=0.49,Y=0.20,Z=11.53
```

*Figure 47 Dynamic behaviour is verified*

Dynamic behaviour is verified by changing the locations and order of nodes. The change in order is detected accurately and data from Source to destination is forwarded successfully.

#### Route Dicovery Results:



*Figure 48 Route Discovered*





## **Chapter 5 – CONCLUSION AND FUTURE WORK**

### **5.1 Conclusion**

This project effectively demonstrates a low-cost, scalable solution to Vehicle-to-Everything (V2X) communication using LoRa and ESP32 modules. It enables real-time data sharing and vehicle dynamic routing, providing priceless contributions to traffic safety and situational awareness regardless of the current communication infrastructure. Utilizing Raspberry Pi and implementing YOLOv8 in Roadside Units (RSUs), the system proposes an intelligent perception layer optimizing congestion recognition and reaction ability. Using light-weight hardware and communication protocols optimized for energy efficiency provides energy-efficient performance that is effective in both urban and rural settings. With this work, a foundation is laid for low-cost intelligent transportation systems (ITS) and how new technologies can be integrated to enable safety-critical use cases while enabling modularity and low costs. In general, the project lays out the possibility of decentralized, infrastructure-independent V2X systems to enable future smart mobility ecosystems.

### **5.2 Future Work**

This would form the basis on which subsequent models would be designed to include more advanced machine learning algorithms to give predictive traffic simulation and vehicle behavior modeling, so as to support proactive decision-making and routing planning. The battery modules would similarly be optimized through energy-aware communication protocols and adaptive duty-cycling mechanisms to ensure working life is maximized, especially in off-grid or remote deployment scenarios. Secondly, the adoption of 5G connectivity would dramatically improve bandwidth and lower the latency of communication, supporting the real-time data exchange at high speeds appropriate to autonomous driving as well as for emergency response. Interoperability with legacy IoT and smart city infrastructure would likewise support other usage scenarios such as environmental sensing, smart traffic lighting, and protection of pedestrians. The modular design of the system places it well today to expand in the future, being adaptable for various geographical conditions and scalable deployment. This opens the possibility of a more integrated and responsive transport network consistent with the vision of fully autonomous and intelligent urban mobility.

## **REFERENCES**

- 1) H. Wang, Y. Zhang, and C. Zhu, "DAFPN-YOLO: An Improved UAV-Based Object Detection Algorithm Based on YOLOv8s," *CMC-Computers, Materials & Continua*, vol. 74, no. 1, pp. 123–136, Apr. 2025. [ResearchGate](#)
- 2) M. C. Lucas-Estañ et al., "An Analytical Latency Model and Evaluation of the Capacity of 5G NR to Support V2X Services using V2N2V Communications," *arXiv preprint arXiv:2201.06083*, Jan. 2022. [arXiv+1arXiv+1](#)
- 3) B. Coll-Perales et al., "End-to-End V2X Latency Modeling and Analysis in 5G Networks," *arXiv preprint arXiv:2201.06082*, Jan. 2022. [arXiv+1arXiv+1](#)
- 4) H. Yu et al., "V2X-Seq: A Large-Scale Sequential Dataset for Vehicle-Infrastructure Cooperative Perception and Forecasting," *arXiv preprint arXiv:2305.05938*, May 2023. [arXiv](#)
- 5) "Benchmark on RPi5 and CM4 running YOLOv8s with RPi AI kit," Seed Studio, 2024. [Online]. Available: [https://github.com/Seeed-Studio/wiki-documents/blob/docusaurus-version/docs/Edge/Raspberry Pi Devices/Raspberry Pi with AI/benchmark on rpi5 and cm4 running yolov8s with rpi ai kit.md](https://github.com/Seeed-Studio/wiki-documents/blob/docusaurus-version/docs/Edge/Raspberry%20Pi%20Devices/Raspberry%20Pi%20with%20AI/benchmark%20on%20rpi5%20and%20cm4%20running%20yolov8s%20with%20rpi%20ai%20kit.md)GitHub
- 6) "Quick Start Guide: Raspberry Pi with Ultralytics YOLOv8," Ultralytics, 2023. [Online]. Available: <https://docs.ultralytics.com/guides/raspberry-pi/Home>
- 7) "An Android-Based Smart RSU Framework for Enhanced Urban Traffic Monitoring," *International Journal of Computer and Information Technology*, vol. 14, no. 2, pp. 45–52, Apr. 2025. [ijcit.com](#)
- 8) "An Experimental Study on the Use of LoRa Technology in Vehicle Communication," *ResearchGate*, Feb. 2021. [Online]. Available: [https://www.researchgate.net/publication/349155098\\_An\\_Experimental\\_Study\\_on\\_the\\_Use\\_of\\_LoRa\\_Technology\\_in\\_Vehicle\\_Communication](https://www.researchgate.net/publication/349155098_An_Experimental_Study_on_the_Use_of_LoRa_Technology_in_Vehicle_Communication)ResearchGate+1Academia+1
- 9) "Feasibility of LoRa-Based Infrastructure-to-Vehicle (I2V) Communication for V2X Applications," *International Journal of Communication Systems*, vol. 34, no. 10, e6002, Oct. 2023. [Wiley Online Library](#)
- 10) "Implementing Cryptography in LoRa-Based Communication Devices for Secure Data Transmission," *SN Applied Sciences*, vol. 3, no. 5, pp. 1–10, May 2021. [SpringerLink](#)1
- 11) "Accurate V2X Traffic Prediction with Deep Learning Architectures," *Frontiers in Artificial Intelligence*, vol. 5, article 1565287, Apr. 2025. [Frontiers+1PMC+1](#)
- 12) "Machine Learning Applications in V2X Networks: A Systematic Review," *Brazilian Congress of Automation*, 2024. [Online]. Available: [https://www.sba.org.br/cba2024/papers/paper\\_900.pdf](https://www.sba.org.br/cba2024/papers/paper_900.pdf)sba.org.br
- 13) "Deep Learning for Predicting Traffic in V2X Networks," *Applied Sciences*, vol. 12, no. 19, article 10030, Oct. 2022. [MDPI+1Frontiers+1](#)
- 14) "Exploring V2X in 5G Networks: A Comprehensive Survey of Location-Based Services," *Vehicular Communications*, vol. 34, article 100456, Jan. 2025. [ScienceDirect](#)
- 15) "Smart City Platform Adoption for C-V2X Services," *EconStor*, 2020. [Online]. Available: <https://www.econstor.eu/bitstream/10419/224845/1/Basaure-Benseny.pdf>EconStor

## ORIGINALITY REPORT

7%	4%	4%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

## PRIMARY SOURCES

1	Submitted to Higher Education Commission Pakistan Student Paper	1%
2	"Advances in Network-Based Information Systems", Springer Science and Business Media LLC, 2018 Publication	< 1%
3	dspace.umh.es Internet Source	< 1%
4	www.coursehero.com Internet Source	< 1%
5	Myoungsu Kim, Insu Oh, Kangbin Yim, Mahdi Sahlabadi, Zarina Shukur. "Security of 6G enabled Vehicle-to-Everything Communication in Emerging Federated Learning and Blockchain Technologies", IEEE Access, 2023 Publication	< 1%
6	d3n9y02raazwpg.cloudfront.net Internet Source	< 1%
7	B. Coll-Perales, M.C. Lucas-Estan, T. Shimizu, J. Gozalvez, T. Higuchi, S. Avedisov, O. Altintas, M. Sepulcre. "Improving the Latency of 5G V2N2V Communications in Multi-MNO Scenarios using MEC Federation", 2022 IEEE	< 1%

# 95th Vehicular Technology Conference: (VTC2022–Spring), 2022

Publication

- 
- |          |                                     |         |
|----------|-------------------------------------|---------|
| <b>8</b> | <b>arxiv.org</b><br>Internet Source | $< 1\%$ |
|----------|-------------------------------------|---------|
- 
- |          |   |         |
|----------|---|---------|
| <b>9</b> | <b>www.hindawi.com</b><br>Internet Source | $< 1\%$ |
|----------|---|---------|
- 
- |           |   |         |
|-----------|---|---------|
| <b>10</b> | <b>Submitted to American University of the Middle East</b><br>Student Paper | $< 1\%$ |
|-----------|---|---------|
- 
- |           |  |         |
|-----------|--|---------|
| <b>11</b> | <b>Franz, Walter, Hartenstein, Hannes and Mauve, Martin [Hrsg.]. "Inter-vehicle-communications based on ad hoc networking principles. The FleetNet Project", Universitätsverlag Karlsruhe, Karlsruhe, 2005.</b><br>Publication | $< 1\%$ |
|-----------|--|---------|
- 
- |           |   |         |
|-----------|---|---------|
| <b>12</b> | <b>Submitted to Universiti Malaysia Perlis</b><br>Student Paper | $< 1\%$ |
|-----------|---|---------|
- 
- |           |  |         |
|-----------|--|---------|
| <b>13</b> | <b>www.edaboard.com</b><br>Internet Source | $< 1\%$ |
|-----------|--|---------|
- 
- |           |   |         |
|-----------|---|---------|
| <b>14</b> | <b>Submitted to King's College</b><br>Student Paper | $< 1\%$ |
|-----------|---|---------|
- 
- |           |   |         |
|-----------|---|---------|
| <b>15</b> | <b>Li, Haipeng. "An adaptive MAC protocol for wireless multi-hop networks with multiple antennas", Proquest, 2011 1108</b><br>Publication | $< 1\%$ |
|-----------|---|---------|
- 
- |           |  |         |
|-----------|--|---------|
| <b>16</b> | <b>U. C. Das, S. K. Verma. "Electromagnetic response of an arbitrarily shaped three-dimensional conductor in a layered earth -- numerical results", Geophysical Journal International, 1982</b><br>Publication | $< 1\%$ |
|-----------|--|---------|

17

Saleh, Muhsin Hassanu. "Energy Efficient Routing Protocols for Underwater Acoustic Wireless Sensor Network.", University of Salford (United Kingdom)

Publication

< 1 %

18

Submitted to University College London

Student Paper

< 1 %

19

Bruno Mendes, Marco Araújo, Adriano Goes, Daniel Corujo, Arnaldo S.R. Oliveira.

"Exploring V2X in 5G Networks: A Comprehensive Survey of Location-Based Services in Hybrid Scenarios", Vehicular Communications, 2025

Publication

< 1 %

20

Honglin Wang, Yaolong Zhang, Cheng Zhu. "DAFPN-YOLO: An Improved UAV-Based Object Detection Algorithm Based on YOLOv8s", Computers, Materials & Continua, 2025

Publication

< 1 %

21

Jan Quadflieg, Gunter Rudolph, Mike Preuss.

"How costly is a good compromise: Multi-objective TORCS controller parameter optimization", 2015 IEEE Conference on Computational Intelligence and Games (CIG), 2015

Publication

< 1 %

22

L. Pautet. "Integrating page replacement in a distributed shared virtual memory", 14th International Conference on Distributed Computing Systems ICDCS-94, 1994

Publication

< 1 %

23 Submitted to Universidad Internacional de la Rioja  
Student Paper < 1 %

---

24 [courses.ece.illinois.edu](https://courses.ece.illinois.edu)  
Internet Source < 1 %

---

25 [openarchive.usn.no](https://openarchive.usn.no)  
Internet Source < 1 %

---

26 [repository.its.ac.id](https://repository.its.ac.id)  
Internet Source < 1 %

---

27 [www.authorea.com](https://www.authorea.com)  
Internet Source < 1 %

---

28 [docplayer.net](https://docplayer.net)  
Internet Source < 1 %

---

29 [dspace.iiuc.ac.bd:8080](https://dspace.iiuc.ac.bd:8080)  
Internet Source < 1 %

---

30 [forum.arduino.cc](https://forum.arduino.cc)  
Internet Source < 1 %

---

31 [forum.seeedstudio.com](https://forum.seeedstudio.com)  
Internet Source < 1 %

---

32 [utpedia.utp.edu.my](https://utpedia.utp.edu.my)  
Internet Source < 1 %

---

33 Adrian I. Petrariu, Alexandru Lavric, Eugen Coca. "LoRaWAN Gateway: Design, Implementation and Testing in Real Environment", 2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME), 2019  
Publication < 1 %

---

34 Mohammed Fadhil, Qutaiba Ibrahim Ali. "An Android-Based Smart RSU Framework for < 1 %

Enhanced Urban Traffic Management",  
International Journal of Computer and  
Information Technology(2279-0764), 2025

Publication

---

35	Submitted to North West University Student Paper	< 1 %
36	dl.ucsc.cmb.ac.lk Internet Source	< 1 %
37	econpapers.repec.org Internet Source	< 1 %
38	unsworks.unsw.edu.au Internet Source	< 1 %
39	Submitted to University of Glasgow Student Paper	< 1 %
40	karczpolska.pl Internet Source	< 1 %
41	www.ijraset.com Internet Source	< 1 %
42	www.maximizemarketresearch.com Internet Source	< 1 %
43	www.mdpi.com Internet Source	< 1 %
44	www.techscience.com Internet Source	< 1 %
45	Anuja Nair, Sudeep Tanwar. "Resource allocation in V2X communication: State-of- the-art and research challenges", Physical Communication, 2024 Publication	< 1 %

---



46 Elias Munapo, Santosh Kumar, Philimon Nyamugure, Trust Tawanda. "Network Optimization: An Introduction to the Network Reconstruction Approach – Network Reconstruction Approach to Optimization", River Publishers, 2025

Publication

< 1 %

47 Mohammad Fardad, Gabriel–Miro Muntean, Irina Tal. "Latency–aware V2X Operation Mode Coordination in Vehicular Network Slicing", 2023 IEEE 97th Vehicular Technology Conference (VTC2023–Spring), 2023

Publication

< 1 %

48 P. Chuychai. "Suppressed Diffusive Escape of Topologically Trapped Magnetic Field Lines", The Astrophysical Journal, 11/01/2005

Publication

< 1 %

49 Soares, André Nogueira. "Edge Computing Image Processing for Contextual Insights Towards Lighting Applications.", Universidade do Porto (Portugal)

Publication

< 1 %

50 [elibrary.tucl.edu.np](http://elibrary.tucl.edu.np)

Internet Source

< 1 %

51 [releia.ifsertao-pe.edu.br](http://releia.ifsertao-pe.edu.br)

Internet Source

< 1 %

52 [repositorio.usp.br](http://repositorio.usp.br)

Internet Source

< 1 %

53 [www.maxlinear.com](http://www.maxlinear.com)

Internet Source

< 1 %

54 Gajendra Kumar Ahirwar, Ratish Agarwal, Anjana Pandey. "Secured Energy Efficient Chaotic Gazelle based Optimized Routing Protocol in mobile ad-hoc network", Sustainable Computing: Informatics and Systems, 2025  
Publication

---

55 L. Van Hoesel, T. Nieberg, Jian Wu, P.J.M. Havinga. "Prolonging the lifetime of wireless sensor networks by cross-layer interaction", IEEE Wireless Communications, 2004  
Publication

---

56 Yan Zhang, Honglin Hu, Masayuki Fujise. "Resource, Mobility, and Security Management in Wireless Networks and Mobile Communications", Auerbach Publications, 2019  
Publication

---

---

Exclude quotes Off

Exclude matches Off

Exclude bibliography Off